Universidad Carlos III de Madrid
Escuela Politécnica Superior

SetList Información de conciertos y discografía

Proyecto Fin de Carrera
Ingeniería Técnica en Informática de Gestión

Autor Israel Martín del Moral
Director D. Daniel Borrajo Millán
Año 2015

Agradecimientos

A todas las personas que me han insistido en que tenía que terminarlo. Y en especial a mi madre a la que se lo debía desde hace 7 años.

Tabla de contenidos

Tabla de Ilustraciones

1 Introducción

Lo primero que haremos en este documento será explicar en qué consiste el proyecto que se ha realizado y los objetivos que se pretenden conseguir con su elaboración.

1.1 Marco proyecto fin de carrera

Este proyecto fin de carrera *Setlist información de conciertos y discografía* extrae de varias páginas web información referente a los conciertos realizados por un artista, así como su discografía, mediante a una serie de wrappers que accederán a cada una de las páginas. Obtendrá distintas informaciones que se mostrarán en una aplicación Android que hará que el usuario pueda consultar los datos de una forma cómoda.

Para la obtención de los setlist o repertorio, se disponía de una única página desde la que extraer la información, pero para la obtención de la discografía, se han analizado distintas páginas para ver cuál era la más completa, con el fin de poder mostrar al usuario el mayor número de artistas y la mayor cantidad de información acerca de ellos.

Los wrappers analizan el código HTML de las distintas páginas utilizadas para mediante la búsqueda de etiquetas ir obteniendo los valores referentes a cada uno de los datos que se quieren tratar en la aplicación.

1.2 Objetivo del proyecto

El objetivo de este proyecto es ofrecerle al usuario en una misma aplicación Android la información de la discografía y los conciertos de sus artistas favoritos sin tener que visitar varias webs para obtener estos datos. Para ello se han diseñado una serie de wrappers que acceden a distintas páginas web para obtener la información y presentársela a usuario de una forma más amigable y que puede ser consultada desde cualquier dispositivo Android.

1.3 Estructura del documento

Esta memoria del proyecto fin de carrera está compuesta por los capítulos que describen a continuación:

- **Capítulo 2. Estado de la cuestión:** En este capítulo se describe el uso de aplicaciones en los dispositivos móviles. Además, se explica cómo se extrae información de páginas web mediante "*web scraping*" mostrando ejemplos de utilización en distintas aplicaciones.

- **Capítulo 3. Objetivos:** Aquí se expone, de forma más amplia que en el punto anterior los objetivos que se pretenden lograr con el desarrollo de este proyecto fin de carrera.

- **Capítulo 4. Trabajo realizado:** En este capítulo se recoge de forma detallada el trabajo realizado para llevar a cabo el proyecto en su resultado final. Se describe cómo se han analizado las distintas páginas de las que se obtiene la información que será usada para mostrarle los datos solicitados al usuario. Se analiza cómo se deben realizar las distintas peticiones a cada uno de los servidores de datos y la forma de obtener la información del código HTML que recibimos como respuesta.

  También se explica todo el proceso de creación de la aplicación Android desarrollada exponiendo cada una de las partes y mostrando aquellos trozos de código que se han considerado relevantes. Y para finalizar se mostrarán los pasos que se siguieron para generar el fichero y poder instalarlo en nuestro dispositivo.

- **Capítulo 5. Resultado:** En este capítulo se describe el funcionamiento de la aplicación de forma que sirva como un breve manual de usuario en el que se describe todas las posibilidades que nos ofrece.

- **Capítulo 6. Conclusiones:** En este capítulo se reflejan las conclusiones sobre las aplicaciones para dispositivos móviles y el acceso a páginas web para obtener información de ellas.

- **Capítulo 7. Futuras líneas de trabajo:** En este capítulo se recogen los futuros pasos que se desean abordar para mejorar las funcionalidades de la aplicación desarrollada.

2 Estado de la cuestión

En la actualidad, mediante nuestros smartphones, tables, smarttv y ahora incluso mediante los nuevos relojes smartwatch estamos continuamente consultando información en internet. Desde nuestros dispositivos podemos usar el navegador para visualizar las distintas páginas web en las que deseamos consultar la información, pero muchas veces las páginas no son muy cómodas de visualizar en los dispositivos debido al tamaño de pantalla de estos puesto que gran parte de ellas no están diseñadas pensando en la navegación desde dispositivos móviles.

Por ello, existen muchísimas aplicaciones que nos permiten visualizar la misma información que está en la web de una forma más cómoda y además nos dan más funcionalidades que las que podemos obtener únicamente usando el navegador.

2.1 Aplicaciones para móviles

En la actualidad, existen innumerables aplicaciones que podemos instalar en nuestros dispositivos móviles que nos permiten hacer casi cualquier cosa que se nos pase por la cabeza. Podemos ver videos, escuchar música, organizar nuestros contactos y agendas, hacer fotos y retocarlas, controlar nuestros entrenamientos cuando salimos a correr o montar en bici, jugar, controlar nuestras redes sociales, encontrar pareja o saber llegar a nuestro destino sin perdernos.

Estas son las aplicaciones de uso más común, pero como hemos dicho nos podemos instalar aplicaciones para hacer casi cualquier cosa que se nos ocurra. Tenemos aplicaciones para anotar a las horas que ha comido nuestro bebé, obtener rutas de senderismo, aprender a reconocer las setas y hongos, guías de viaje, controlar ronquidos con la posibilidad de hacer gráficas, crear memes graciosos e incluso hacernos un *selfie* y componer una foto en la que aparezcamos con el pequeño Nicolás.

Con el uso de las aplicaciones móviles ahora muchos aparatos que eran de uso habitual que ya no imprescindibles. Hemos dejado de lado los reproductores MP3, el GPS, el despertador, las agendas, el podómetro y la linterna. Ahora todas esas cosas ya las llevamos en nuestros teléfonos y las podemos utilizar con diversas aplicaciones que nos hacen sacar el mayor partido posible de ellas.

Por lo tanto, nuestro Smartphone o tablet se ha convertido en un potentísimo ordenador en tamaño reducido, aunque cada vez menos, que llevamos en el bolsillo y que nos permite hacer casi de todo incluso hablar por teléfono.

Existen multitud de aplicaciones, pero ahora nos referiremos a aquéllas que traen datos de internet, los procesan y se los presentan al usuario de una forma más estética y manejable.

Entre las aplicaciones de este tipo, nos encontramos aplicaciones con las que podemos consultar el tiempo que hará en los próximos días, ver a qué hora pasara nuestro autobús con la posibilidad de tener almacenados los números de las paradas que solemos frecuentar, consultar los resultados de nuestros deportes favoritos teniendo la posibilidad de que nos informe en tiempo real de las incidencias que sucedan en los partidos de nuestro equipo favorito.

Muchas de estas aplicaciones son oficiales y están diseñadas por el propio proveedor de datos de la página web, el cual puede obtener la información de sus bases de datos de la misma manera que lo haría para mostrar la información en la web. Pero hay otras muchas aplicaciones lo que hacen es consultar los datos de la web del proveedor para obtener de ella la información que considere necesaria y usarla para mostrársela al usuario en su aplicación de una forma más visual que la que suele aparecer en la web visualizada desde el navegador.

Como ejemplo vamos a mostrar cómo se visualiza la información referente al tiempo de espera en una parada determinada de los autobuses de la EMT de Madrid. En la figura 2-1 se muestra cómo lo veríamos directamente desde el navegador en nuestro smartphone.

Figura 2-1 Búsqueda en navegador de parada y tiempo de espera

Como vemos la web no tiene un diseño muy adecuado para trabajar desde un dispositivo móvil puesto que está diseñada para ser visualizada en un PC y esto hace que desde un smartphone el tamaño sea muy reducido. Por ello, para acceder a esta información es más cómodo trabajar con alguna de las aplicaciones que nos proporcionan información a partir de este servicio de la EMT.

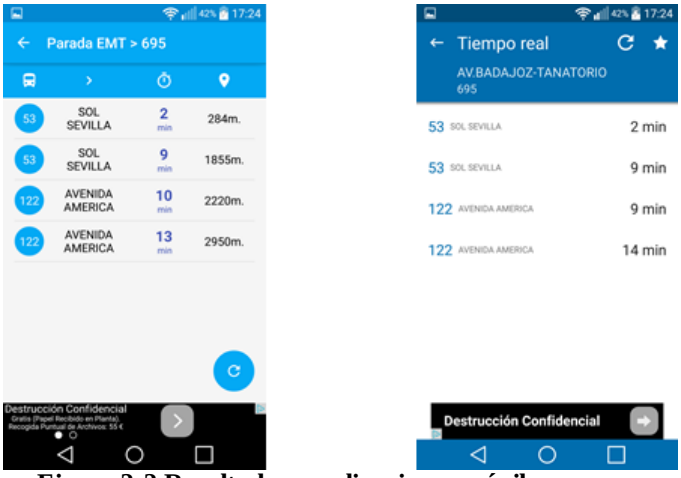

Figura 2-2 Resultado en aplicaciones móviles

Como podemos ver en la figura 2-2 que muestra la representación de la información que nos ofrecen distintas aplicaciones que se encuentran disponibles en el Market de Android, se nos muestran los mismos datos que aparecían en la web pero de una forma bastante más amigable. Además, estas aplicaciones nos permiten la posibilidad de almacenar en nuestro terminal nuestras paradas favoritas y otras funcionalidades que no tenemos en la web.

Estas aplicaciones, han sido creadas por usuarios ajenos a EMT, por lo tanto para obtener los datos para responder a la consulta realizada por el usuario tienen que acceder a la página web http://www.emtmadrid.es/ y realizar consultas a la página y analizar la información devuelta por está para mostrarle la información al usuario. Este mismo proceso es el que se hace en nuestra aplicación y recibe el nombre de *web scraping* (rascado).

2.2 Web scraping

A continuación se expone en que consiste el *web scraping*, ejemplos de uso y los posibles inconvenientes que nos podemos encontrar.

2.2.1 Definición

El *web scraping*, que traducido de forma literal al castellano seria "rascado", es la técnica utilizada para extraer información de la web mediante la simulación de la navegación de una persona mediante la utilización de programas diseñados para tal efecto.

La finalidad del *web scraping* es poder obtener aquella información que nos interese de sitios web para su posterior utilización por nuestros propios programas. Consiste en ir extrayendo aquellas partes que necesitamos de las páginas HTML, que generalmente están poco estructuradas, para obtener la información que consideramos relevante, de tal forma que la podamos procesar con posterioridad, bien sea para almacenarla en nuestras bases de datos o simplemente mostrársela al usuario de una forma amigable.

A la hora de realizar *web scraping* para extraer la información que deseamos obtener de un determinado sitio web existen varias técnicas, algunas de las cuales pasamos a exponer a continuación: [wiki01]

- «Copiar y pegar» humano: en ocasiones la única forma que tenemos de obtener la información de un sitio web es de forma manual, seleccionando la información en la página de origen para copiarla y posteriormente pegarla en donde la necesitemos usar, por ejemplo, en el procesador de texto o en una hoja de cálculo, Esta opción es útil cuando el sitio al que accedemos tiene implementadas medidas para evitar que desde programas se pueda obtener de una forma automática la información. De este tipo de medidas "*anti-scraping*" hablaremos más adelante.

- Protocolo HTTP: mediante peticiones GET y POST del protocolo HTTP podemos acceder a páginas webs estáticas y dinámicas para obtener el contenido.

  Aquellos sitios web que reciben consultas del usuario y devuelven la información en función de los datos almacenados en sus bases de datos, por lo general, nos muestran páginas en las que los datos tienen la misma estructura, por lo cual es posible mediante un programa ir extrayendo información de la web utilizando la búsqueda por etiquetas. Esta ha sido la técnica que se ha usado para el desarrollo de la aplicación y que explicaremos más adelante cómo se ha realizado.

- *Parsers* de HTML: Con lenguajes como XQuery y HTQL se puede ser analizar documentos, para recuperar y transformar su contenido del código HTML para obtener la información que necesitamos.

- Aplicaciones para *web scraping*: existen en el mercado aplicaciones que podremos utilizar para realizar *web scraping*. Estas aplicaciones pueden reconocer automáticamente la estructura de una determinada página y le proporciona al usuario una interfaz desde la cual podrá seleccionar los campos que son de interés dentro del documento. Con ello nos ahorramos el trabajo de tener que desarrollar aplicaciones para realizar dichas búsquedas.

- Reconocimiento de información semántica: en el caso de que las páginas a las que queramos acceder incluyan metadatos o cierta información semántica como anotaciones o comentarios, nos permiten recuperar estos metadatos y extraer la información que deseamos a partir de la estructura que hemos obtenido.

Uno de los mayores problemas que presenta la realización del *web scraping*, y el cual hemos sufrido en un par de ocasiones a lo largo del desarrollo de este proyecto, es que el sitio web realice cambios sobre el diseño de las páginas que lo componen. Estos cambios, puede ser que afecten a las etiquetas que nosotros tenemos identificadas para saber qué parte de la información necesitamos obtener.

En el caso de que las partes de la página a las que accedamos hayan sido modificas tendremos que volver a realizar el análisis de la misma para que podamos seguir obteniendo la misma información que extraíamos anteriormente.

2.2.2 Ejemplos de utilización

Además del ejemplo de la aplicación que nos proporcionaba información del tiempo de espera en las paradas de la EMT, a continuación paso a exponer otros ejemplos de la utilización de *web scraping* y su aplicación en nuestra vida cotidiana.

2.2.2.1 Meta buscadores de viajes

Hoy en día, vemos en televisión muchos anuncios de buscadores de vuelos y hoteles por internet. Páginas como www.trivago.es, www.kayak.es o www.trabber.es, nos llaman la atención con eslóganes del tipo "Compara entre miles de ofertas y encuentra tu hotel ideal".

Utilizando la página www.trabber.es como ejemplo vamos a proceder a explicar cuál es la metodología empleada para realizar la petición de información a un sitio web y su posterior análisis y tratamiento de la información para sacar de ella el máximo rendimiento posible.

Este tipo de páginas, son un claro ejemplo de lo que se puede hacer mediante el *web scraping*. Como podemos ver en la figura 2-3, en el formulario del buscador seleccionaremos cuáles serán nuestros criterios de búsqueda y será la propia página la que comenzará a lanzar llamadas a los distintos operadores con estos criterios.

Figura 2-3 Búsqueda en www.trabber.es

Lo primero que debe conocer, es como construir la llamada al sitio web para poderle enviar nuestra consulta. Para la realización de las llamadas, se utiliza el protocolo de

comunicaciones HTTP. Este protocolo nos ofrece métodos para realizar llamadas a un determinado sitio web, que son GET y POST. Ambos son métodos de petición y respuesta, pero entre ellos existen algunas diferencias. *[micayael]*

- GET sirve para obtener información mientras que POST se utiliza para enviar información. Es decir, cuando enviamos un método GET, lo único que esperamos obtener del servidor es una respuesta con información sin necesidad de que los datos sean procesados, por ejemplo cuando pulsamos un enlace en una página.

- El método POST sirve para enviar información al servidor para que la procese y nos devuelva una respuesta. Este es el caso de los formularios en los que la petición lleva una serie de parámetros que es la información que nosotros le enviamos al servidor para que sea procesada.

A continuación, vamos a mostrar como seria la consulta a uno de los sitios web a los que accede, en este caso http://www.tripsta.es. Esta página utiliza un formulario para introducir la información, por tanto, se deberá lanzar en una llamada con el método POST. Para componer la llamada, se deberá conocer el formato que espera recibir el servidor y para ello los más cómodo es rellenar en el navegador el formulario, pulsar el botón enviar datos y analizar la dirección para ver cuáles son parámetros son los que se le pasan. Este sería un ejemplo de dirección.

http://www.tripsta.es/vuelos-baratos/resultados?dep=Madrid+-+Madrid-Barajas+%28MAD%29%2C+Espa%C3%B1a&arr=Berl%C3%ADn+-+Todos+los+Aeropuertos+%28BER%29%2C+Alemania&isRoundtrip=1&obDate=01%2F11%2F2015&ibDate=10%2F11%2F2015&obTime=&ibTime=&extendedDates=0&resetStaticSearchResults=1&passengersAdult=2&passengersChild=0&passengersInfant=0&airlineCode=&class=&directFlightsOnly=0

Ahora que ya conoce cómo se debe realizar la llamada al servidor hay que componer esa llamada a partir de los parámetros de entrada que se desean y enviarla al servidor. En este caso la página trabaja con 71 servidores por lo que realizará 71 llamadas distintas y quedará a la espera de la información que le devuelva cada una de ellas.

El buscador, en el momento que recibe las respuestas de las distintas páginas a las que accede, comienza a realizar *web scraping* sobre el código HTML de la respuesta para obtener toda la información que se considera necesaria acerca de los vuelos para utilizarla con posterioridad.

Para que el buscador pueda realizar este análisis, previamente los desarrolladores han estudiado la estructura de la página de resultados que devuelve cada uno de los operadores de vuelos, analizando el código fuente. Esta tarea la explicaremos en el capítulo Trabajo realizado mostrando como se ha realizado el análisis de las páginas a las que accede nuestra aplicación.

Una vez que el servidor de *Trabber* tiene la respuesta de todos los servidores a los que accede y ha obtenido la información que necesita de cada una de ellas, ya está preparado para mostrárnosla tal y como podemos ver en la figura 2-4.

Figura 2-4 Resultado búsqueda en Trabber.

A partir de este momento, el servidor nos permite interactuar con él indicándole los filtros que queremos poner para que nos muestre la información, como por ejemplo, número de escalas, hora de salida y llegada, rango de precios. Toda estas operaciones ya las puede realizar de forma inmediata y mostrárnoslas porque dispone de la información de todos los sitios a los que accede.

Ahora ya solo nos queda seleccionar el vuelo que mejor se ajusta a nuestras necesidad y pulsar sobre el enlace y la página nos redirija a la web de la operadora ya con los datos del vuelo que hemos seleccionado para que continuemos con el proceso de compra.

La ventaja que nos ofrece este tipo de buscadores ya no es solo que sean capaces de buscar por nosotros en gran cantidad de sitios a la vez, con el ahorro de esfuerzo y tiempo que supone, sino que además, al ya tener almacenada en sus bases de datos la información de las consultas realizadas por otros usuarios o de forma automática por servidor, nos pueden ofrecer otras posibilidades de explotar la información. Por ejemplo, permiten la posibilidad de ofrecernos vuelos desde nuestra ciudad en una fecha y un rango de precios sin especificar el destino. Esto lo realiza de forma inmediata puesto que no tienen que ir a la web de las operadoras porque ya tienen los datos en el servidor.

2.2.2.2 Análisis del mercado inmobiliario

Otro ejemplo de utilidad del *web scraping*, está relacionado con una empresa americana que se dedicaba a buscar en internet para obtener información de distintas páginas web en las que aparece información de venta de casas. *[youtube02]*

La información obtenida la almacenan en sus bases de datos para posteriormente calcular el precio del metro cuadrado en las distintas zonas de EEUU. Cuando detectan que existe alguna propiedad que está por debajo del precio medio de mercado de su zona, analizan la viabilidad de su compra para su posterior reforma y venta, obteniendo con ello un beneficio económico.

2.2.2.3 Cartera de valores

Otro ejemplo de uso de *web scraping* es la capacidad de leer datos en formato tabla que nos proporcionan aplicaciones como la hoja de datos Microsoft Excel. Mediante la opción Datos desde Web nos da la opción de realizar una consulta sobre una página web para obtener de ella la tabla seleccionada para su posterior tratamiento en Excel. *[school]*

Como ejemplo, vamos a ver cómo podríamos obtener la cotización del Ibex 35 para poder analizar en Excel nuestra cartera de valores. Lo primero que tenemos que hacer es ir a la pestaña Datos y el botón Desde Web y poner la url desde la que deseamos obtener la tabla con la cotización de las acciones y tendríamos el resulta que se muestra en la figura 2-5.

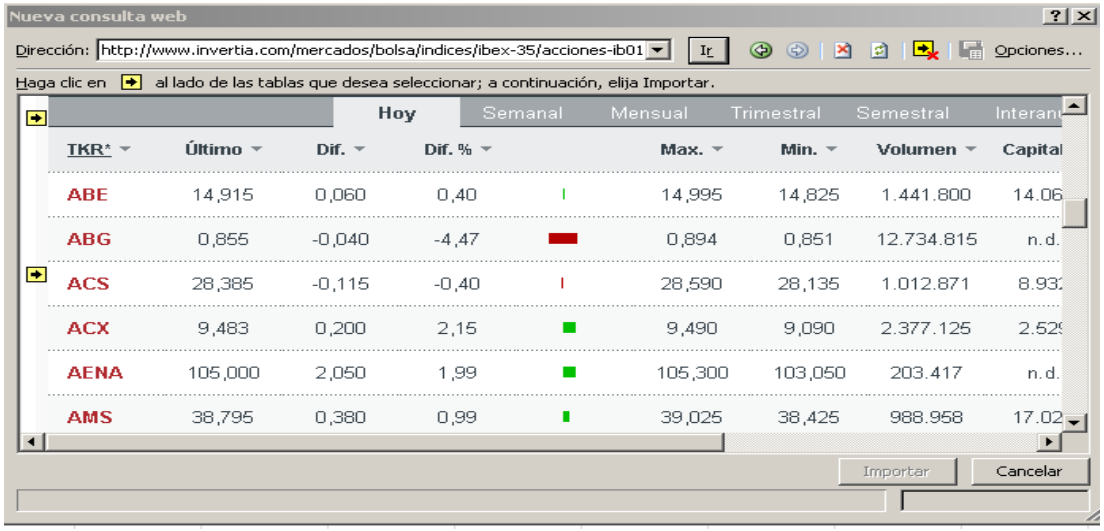

Figura 2-5 Selección de tabla en consulta Web

Entre las varias tablas que puede reconocer Excel dentro de la página web, deberemos marcar cual es la que nos interesa y pulsar el botón Importar. De esta forma, tendremos en una hoja de Excel una tabla con toda la información de los valores del IBEX35 [Figura 2-6]. Ahora, podemos trabajar con estos datos para realizar los cálculos que consideremos necesarios. De esta forma ya tenemos controlar al instante la rentabilidad de nuestra cartera de valoras puesto que en todo momento podremos actualizar la consulta que realizamos a la web con las cotizaciones.

A1

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	TKR*	Último	Dif.	Dif. %		Max.	Min.	Volumen	Capital	Rt/Div	BPA	PER	Hora**
2	ABE	14,915	0,06	0,4	GráficoGráfico	14,995	14,825	1.441.800	14.068	4,43%	1,78	6,77	17:35
3	ABG	0,855	-0,04	-4,47	GráficoGráfico	0,894	0,851	12.734.815	n.d.	13,22%	0,09	19,82	17:35
4	ACS	28,385	-0,115	-0,4	GráficoGráfico	28,59	28,135	1.012.871	8.932	4,05%	1,29	12,16	17:35
5	ACX	9,483	0,2	2,15	GráficoGráfico	9,49	9,09	2.377.125	2.529	4,73%	0,24	26,14	17:35
6	AENA	105	2,05	1,99	GráficoGráfico	105,3	103,05	203.417	n.d.	n.a.	1,84	n.a.	17:35
7	AMS	38,795	0,38	0,99	GráficoGráfico	39,025	38,425	988.958	17.024	1,80%	n.a.	n.a.	17:35
8	ANA	70,5	-0,21	-0,3	GráficoGráfico	70,94	69,76	194.878	4.037	2,84%	1,8	17,63	17:35
9	BBVA	7,894	-0,02	-0,25	GráficoGráfico	7,945	7,826	23.205.116	49.774	4,69%	0,44	13,43	17:35

Figura 2-6 Excel con cotizaciones

2.2.2.4 Web Scraper en Google Chrome

El navegador Google Chorme nos proporciona una extensión que se llama Web Scraper, la cual nos permite hacer *scraping* de forma sencilla desde el navegador. Lo único que tenemos que hacer para poder trabajar con ella, es instalarnos la extensión y ya podremos analizar cualquier página únicamente con pulsar el botón derecho del ratón sobre el elemento que queramos estudiar.

Ahora vamos a ver cómo funciona Web Scraper. Para ello, utilizaremos como ejemplo la página http://www.setlist.fm/ que es una de las que obtiene datos la aplicación desarrollada. Lo único que tenemos que hacer es pulsar con el botón derecho sobre el objeto que queremos analizar. En nuestro caso lo haremos sobre el nombre del concierto tal y como podemos ver en la figura 2-7.

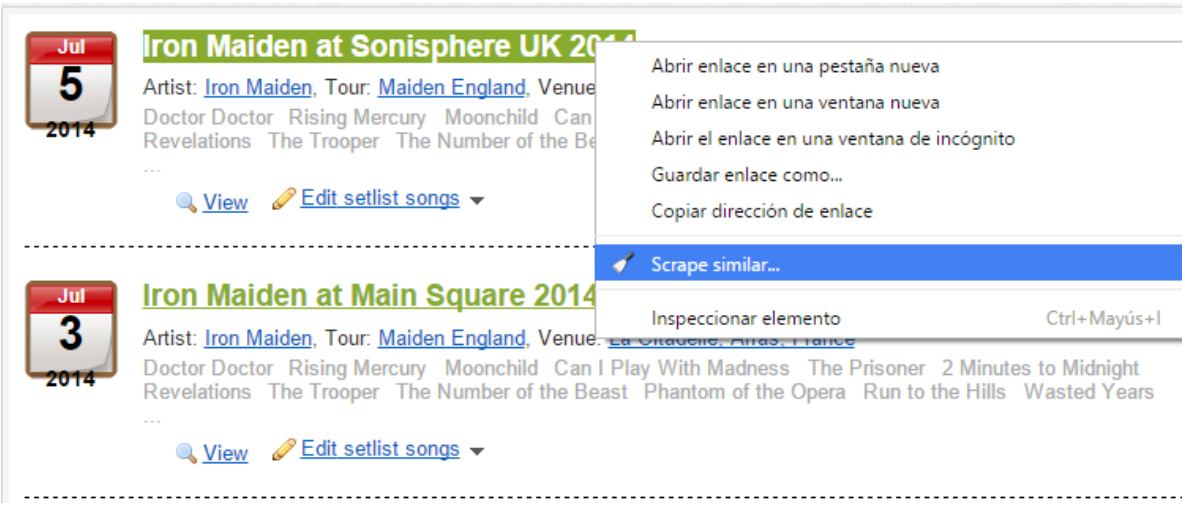

Figura 2-7 Uso de Web Scraper

Esto hará que se nos abra una ventana en la que nos aparecen todas las entradas similares que hay en la página a la que habíamos seleccionado. En nuestro caso, será el nombre de todos los conciertos ofrecidos por el grupo. [Figura 2-8]

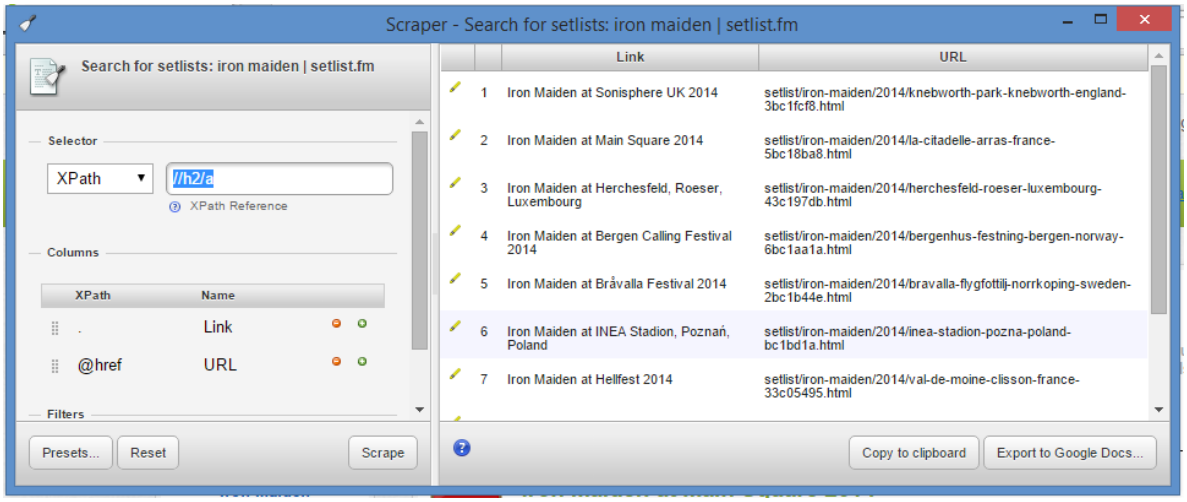

Figura 2-8 Resultado Web Scraper

Los resultados obtenidos los podremos copiar al portapapeles para utilizarlos en donde los necesitemos. De esta forma, hemos hecho de una forma sencilla *web scraping* son una página.

2.2.3 Sistemas de bloqueo

Pero no siempre es posible acceder a todos los sitios web para hacer *web scraping* o simplemente interactuar con él mediante la realización de peticiones GET o POST de forma automática, puesto que el administrador puede que no quiera permitir que se realicen accesos a sus páginas sin que estos sean realizados directamente por una persona desde un navegador.

Los motivos por los que un sitio web pude que no desee que se acceda a él mediante peticiones realizadas de forma automática son los que se describen a continuación: *[wiki01]*

- Evitar que el sitio web pueda ser colapsado mediante la realización de llamadas continuas desde una aplicación que pudiera hacer que la página se colgara al no ser capaz de resolver tantas peticiones de servicio.

- En páginas en las que por ejemplo se realiza un alta de usuario en su BBDD se quiere evitar que desde una aplicación externa se puedan crear un gran número de usuarios ficticios en la BBDD que lo único que haría es tener basura en las tablas, con los problemas que habría de sobredimensionamiento y rendimiento de la base de datos.

- También podría ser que el propietario del sitio web no quisiera que su información sea explotada de forma automática desde otras aplicaciones y que solo se pueden visualizar los datos desde el navegador confirmando que es una persona y no un robot de búsqueda.

Para evitar esta clase de accesos el administrador del sitio web puede utilizar varias técnicas que o bien bloqueen totalmente el acceso para realizar peticiones o disminuir su impacto.*[wiki01]*

- Bloquear dirección IP, en el caso de que se detecte un gran número de peticiones desde una determinada dirección IP se puede bloquear el acceso al sitio web desde dicha dirección.

- Añadir entradas al fichero robots.txt. de esta forma los robots de búsqueda pueden ser detenidos.

- Incrementar el uso de JavaScript y AJAX, que hace que sea más difícil realizar *web scraping* y simular que estamos trabajando desde un navegador.

- Servicios comerciales *antibots*. Algunas empresas ofrecen servicios *antibots* y *antiscraping*.

- Añadir un *captcha* u otro sistema de verificación manual al sitio web. Con ello, no siempre garantizamos el completo bloqueo de los *scrapers*, pero mediante esta técnica se dificulta el acceso de los mismos a los sitios webs. Un *captcha* es esa imagen que nos encontramos en muchos sitios web que nos obliga a meter el texto que aparece normalmente tachado o distorsionado. En la figura 2-9 podemos ver un ejemplo. *[genbeta]*

Figura 2-9 Ejemplo de captcha

El motivo por el que la imagen se muestra de esa manera es evitar que los programas OCR de reconocimiento de caracteres puedan ser capaces de interpretar de forma correcta lo que aparece como letras. Pero como siempre, ya hay medios para ser capaces de reconocer dichos textos. Lo que hacen es limpiar la imagen de los filtros de color o las líneas que la tachan, segmentar la cadena y analizar cada una de las letras por separado.

Este sistema es un grave problema para las personas con deficiencia visual a la hora de navegar por las páginas que los incluyen puesto que los lectores de pantalla (Jaws, Zootext) no pueden interpretarlos, siendo esto una barrera que hace imposible continuar navegando por dichas páginas.

Google ha sacado un nuevo sistema llamado *reCAPTCHA* en el que únicamente tenemos que marcar un *check* en el que indicamos que no somos un robot, y Google analiza ver cómo ha interactuado el usuario con el *captcha* y mediante un análisis de riesgo decidir si se trata de un humano o un robot. Con este sistema se cubre mejor el problema de la accesibilidad. En la figura 2-10 podemos ver un ejemplo de reCAPTCHA.

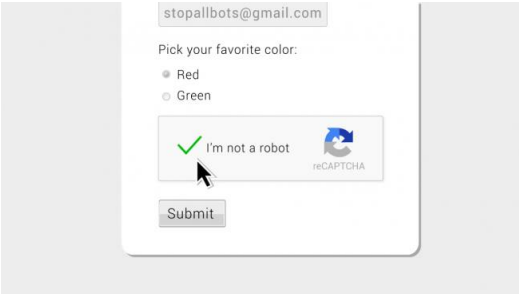

Figura 2-10 Ejemplo reCAPTCHA

2.2.4 Cuestiones legales

En muchas ocasiones el *web scraping* va en contra de las condiciones del sitio web al que estamos accediendo, aunque muchas veces estos términos no están lo suficientemente claros. En Estados Unidos la corte dictó en el caso Feist Publications v. Rural Telephone Service que la duplicación de hechos es permitida. Las cortes de Estados Unidos en ciertas ocasiones han reconocido que ciertos usos de los *scrapers* no deberían estar permitidos. Podría considerarse una computadora como una propiedad personal, y de esta forma el *scraper* estaría entrando sin autorización en esta propiedad. En el caso más conocido, eBay vs Bidder's Edge, la segunda empresa tuvo que parar de realizar peticiones automáticas al sitio de eBay. En este caso, Bidder's Edge pujaba automáticamente por ciertos productos en este sitio. *[wiki01]*

En la actualidad se está tendiendo más a que los jueces protejan a los sitios web para que sus datos no sean utilizados sin su consentimiento, aunque muchas veces depende del tipo de datos que se obtengan mediante *web scraping* y el uso que se les dé.

3 Objetivo

Adquirir los conocimientos necesarios para poder realizar aplicaciones Android para dispositivos móviles. Para ello, se consultaran distintos manuales y videos para adquirir los conocimientos básicos. Además, se buscará ejemplos específicos de aquellos controles y funcionalidades que serán utilizados durante el desarrollo de la aplicación, consultando en foros aquellos problemas que se presenten y que ya hayan sido resueltos por otros usuarios.

Desarrollar una aplicación que nos permita consultar, desde un entorno amigable y fácil de utilizar, la discografía de nuestros artistas favoritos, así como la posibilidad ver que canciones han interpretado en conciertos anteriores. Además, para cada una de las canciones de la discografía o del repertorio podremos visualizar su video en YouTube a la vez que leemos la letra.

4 Trabajo realizado

4.1 Introducción

Para la realización de este proyecto, lo primero fue aprender las nociones básicas de Android. Siguiendo algunos tutoriales *[sgoliver]* y videos de Internet *[youtube01]*, conseguí realizar una primera aplicación para que pudiera ser ejecutada en un smartphone y tablet. A partir de este punto, se comenzó a desarrollar las distintas partes que componen la aplicación.

Puesto que el lenguaje utilizado para el desarrollo de esta aplicación Android ha sido Java *[Naughton]*, del cual ya tenía algún conocimiento, lo siguiente era conseguir en Internet ejemplos de cada una de las partes que se debían realizar y buscar en foros respuestas a los distintos problemas que se iban presentando durante el desarrollo.

Como ya hemos comentado, la aplicación consta básicamente de dos partes. Por un lado, está la obtención de los conciertos. Para ello hemos utilizado la web www.setlist.fm , la cual ya utilizábamos con anterioridad, conociendo las distintas posibilidades que nos ofrecía y las cuales se ajustaban a los datos que deseábamos plasmar en la aplicación.

A la hora de obtener la discografía y las letras de las canciones, la tarea fue un poco más complicada porque algunas páginas nos daban esta información, como por ejemplo www.lyricsmania.com , pero no nos proporcionaba información de algunos de los grupos que buscábamos, o dicha información era escasa, por lo que optamos por www.coveralia.com . Dicha página, tiene mayor cantidad de datos de artistas, y además, nos proporcionaba la portada de los discos, lo cual le aporta a la aplicación un extra que le hará más atractiva.

Ya que teníamos las páginas de las que obtener los datos, las tareas que hubo que realizar fueron las siguientes:

- Petición HTTP: Realizando una búsqueda desde la web en el explorador obtuvimos el formato con el que teníamos que componer la url para que nos proporcionara la información que deseábamos buscar.

- Análisis del código HTML: Con la función *Inspeccionar elemento* o *Ver código fuente de la página* del navegador fuimos analizando cuáles serían las que etiquetas que se debían utilizar para realizar la búsqueda de la información que deseábamos extraer de cada una de las páginas a las que accedíamos.

Por ejemplo, al buscar en *http://www.coveralia.com* la discografía de Iron Maiden nos dará que la discografía se encuentra en la siguiente URL *http://www.coveralia.com/discografias/iron-maiden.php*

Analizando la URL, vemos que tiene una parte común para que todos los artistas, que seria http://www.coveralia.com/discografias/, y concatenado con ella el nombre del artista separado por guiones, en nuestro caso iron-maiden y para finalizar tendría '.php'. Por ello, cuando queremos obtener la página HTML con la discografía del artista a buscar compondremos dicha dirección y del *string* obtenido con el código de la página comenzaremos a buscar la parte que nos interesa para obtener la información correspondiente a cada uno de los discos.

Este sería un ejemplo de peticiones realizando una llamada al método GET del protocolo HTTP puesto que aunque en un primer momento le pasamos al nombre del artista como parámetro, vemos que la dirección lleva en la url el nombre del artista, por lo tanto, yo es necesario que la llamemos utilizando el nombre como parámetro.

En cambio ahora veremos cómo la llamada a la página www.setlist.fm para obtener la discografía de Iron Maiden sí es una llamada al método POST puesto que cuando realizamos la búsqueda en nuestro navegador vemos que la página resultado se compone de la url de la página de búsqueda más una lista de parámetros que especifican la búsqueda que deseamos realizar.

http://www.setlist.fm/search?query=artist:%28iron+maiden%29+city:%28Madrid%29

En este caso esta es la dirección que deberemos hacer en el caso de que queramos obtener los conciertos realizados por Iron Maiden en la ciudad de Madrid.

4.2 Modelo de datos

La aplicación tiene 4 clases principales en las que se almacenan los datos obtenidos de los distintos sitios web a los que accede. Dichas clases se relacionan entre sí tal como se muestra en la figura 4-1.

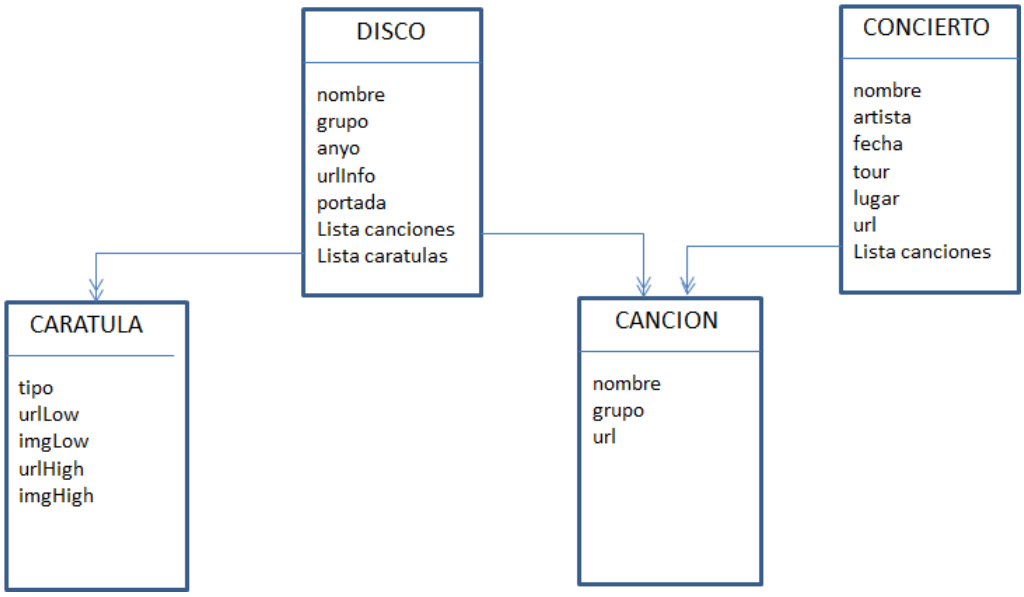

Figura 4-1 Modelo de datos

A continuación, pasamos a describir la finalidad y los datos que almacenan cada una de las clases y cuál es relación con el resto de las clases del modelo.

- Canción: En esta clase se recoge el título de la canción, el artista al que pertenece y la url dentro del servidor de coveralia en donde se encuentra la letra. Esta clase se utiliza en una lista en las clases Concierto y Disco en donde se almacenarán el repertorio del concierto y las canciones que componen el disco respectivamente.

- Caratula: Esta clase contiene el tipo de imagen (portada, CD, Trasera) así como la url de la página y el archivo en la que podemos obtenerla. Estos dos últimos atributos se repiten dos veces ya que almacenamos la información de la imagen en baja y alta calidad. Esta clase se utiliza en la clase Disco para guardar la relación de imágenes del disco que tenemos. En la actualidad únicamente trabajamos con el tipo portada.

- Disco: En esta clase se recoge el nombre del disco, su año de publicación, el artista al que pertenece, la url correspondiente a la imagen de la portada en formato de baja calidad, una lista de objetos de la clase Canción con la información de las canciones que componen el disco y otra lista de objetos de la clase Caratula con la lista de las distintas imágenes disponibles.

- Concierto: para esta clase se almacenan el nombre, el artista, la fecha en la que tuvo lugar, la gira a la que pertenece, el lugar donde se celebró, la url dentro del servidor de setlist.fm que contiene la lista de canciones que fueron interpretadas en dicho concierto y una lista de objetos de la clase Canción con información de cada una de las canciones.

Además de estas cuatro clases, se ha creado la clase videoItem que utilizaremos de forma puntual para recoger la información de cada uno de los videos que nos devolverá la consulta al API de YouTube.

Para cada una de estas clases, se han creado los métodos set y get para escribir y leer cada uno de los atributos de la clase. Además de estos, se han creado otros métodos que harán que nuestra clase sea *Parceble* lo que nos permitirá que pasar objetos de las clases que hemos definido entre las distintas actividades de la aplicación puesto que si no hacemos esto, Android solo nos permite pasar como parámetros los tipos básicos.

```
@Override
public int describeContents() {
	// TODO Auto-generated method stub
	return 0;
}
@Override
public void writeToParcel(Parcel dest, int flag) {
	// TODO Auto-generated method stub
	dest.writeString(nombre);
	dest.writeString(grupo);
	dest.writeString(anyo);
	dest.writeString(urlInfo);
	dest.writeString(portada);
	dest.writeTypedList(canciones);
	dest.writeTypedList(caratulas);}
```

```
public Disco(Parcel in) {
        this.nombre = in.readString();
        this.grupo = in.readString();
        this.anyo = in.readString();
        this.urlInfo = in.readString();
        this.portada = in.readString();
        in.readTypedList(canciones, Cancion.CREATOR);
        in.readTypedList(caratulas, Caratula.CREATOR);}
@SuppressWarnings("unchecked")
public static final Parcelable.Creator<Disco> CREATOR = new Parcelable.Creator<Disco>() {
        public Disco createFromParcel(Parcel in) {
                return new Disco(in);       }
        public Disco[] newArray(int size) {
                return new Disco[size];     }
};
```

4.3 Desarrollo de la aplicación

A continuación se describe el desarrollo y funcionamiento de la aplicación realizada. Mediante imágenes de las web a las que se accede, pantallazos de cómo queda la aplicación en nuestro smartphone y trozos del código realizado, se explicara cómo se ha realizado el proceso de extracción de información y la programación de las distintas clases que han llevado al resultado final obtenido.

En un primer momento comenzamos a programar utilizando Eclipse pero mediado el proyecto decidimos cambiar a Android Studio puesto que Eclipse nos estaba dando problemas con la instalación de algunos componentes y por ello decidimos cambiar de entorno de programación.

Para poder realizar aplicaciones Android debemos tener instalado el Android SDK (SoftwareDevelopment Kit) que incluye un conjunto de herramientas de desarrollo. Está compuesto por un depurador de código, librerías, un emulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales.

Lo primero que tuvimos que hacer es crear un nuevo proyecto Android en nuestro entorno de programación de Android Studio que directamente me nos creó la estructura de carpetas y ficheros que componen la aplicación [Figura 4-2] y que a lo largo de este capítulo iremos explicando para qué se utiliza cada una de ellas.

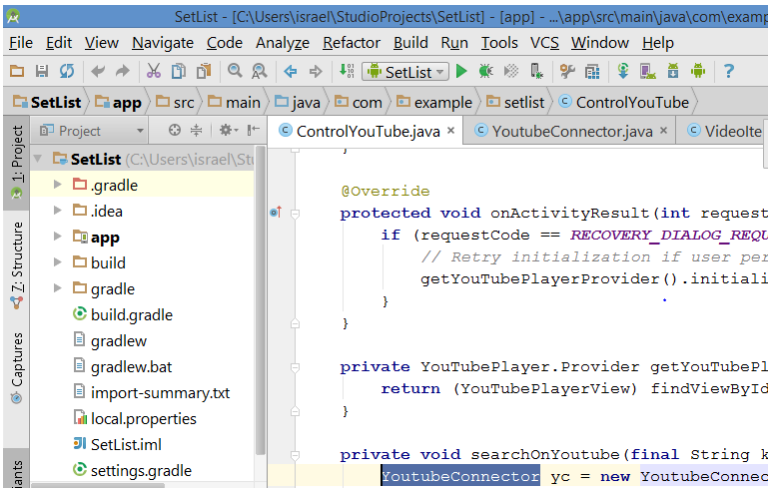

Figura 4-2 Estructura del proyecto Android

El primero de los ficheros que vamos a explicar es de nuestra aplicación Android es AndroidManifest.xml, en él se realiza la configuración del proyecto. En primer lugar, se deben definir los distintos permisos que tendrá nuestra aplicación y que se le mostraran al usuario a la hora de realizar la instalación del archivo APK en su smartphone o tablet. En nuestro caso, estos permisos serán al acceso a Internet (android.permission.INTERNET) y la comprobación del estado de la red (android.permission.ACCESS_NETWORK_STATE)

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Aparte de los que se ha indicado en nuestra aplicación, Android tiene entre otros permisos poder escribir ficheros en nuestro terminal, utilizar nuestra ubicación, uso de la cámara o poder realizar llamadas, entre otros.

Otra de las cosas que se definen en el fichero AndroidManifest.xml es cuál será la actividad que se ejecutará al iniciar la aplicación, que en nuestro caso, será MainActivity

```
<activity android:name="com.example.setlist.MainActivity" android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

Desde esta actividad inicial, la aplicación irá llamando al resto de las actividades de la aplicación. Todas estas actividades, también deberán estar definidas en este fichero para que después puedan ser llamadas.

```
<activity android:name="com.example.setlist.ControlConciertos" android:label="Conciertos" />
<activity android:name="com.example.setlist.ControlCanciones" android:label="Canciones" />
<activity android:name="com.example.setlist.Letra" android:label="Letra" />
<activity android:name="com.example.setlist.ControlDiscos" android:label="Discos" />
<activity android:name="com.example.setlist.ControlPortada" android:label="Portada" />
<activity android:name="com.example.setlist.ControlYouTube" android:label="Video" />
```

Lo primero que nos presentará nuestra aplicación al ser ejecutada será una pantalla en la que a partir de la introducción del artista que deseamos consultar, podremos obtener información de su discografía o de sus conciertos realizados.

La visualización de la estructura de la pantalla que se mostrará se hace mediante ficheros xml en los que se definen los elementos que componen la pantalla y cada uno de los atributos de estos elementos. En nuestro caso el fichero de la MainActivity será activity_main.xlm.

Puesto que las aplicaciones Android pueden ser ejecutadas en distintos tipos de dispositivos y estos pueden tener distintas resoluciones, Android nos da la posibilidad de una estructura de carpetas en la que podremos meter distintas versiones del archivo xml de la definición del layout el cual será usado en función de la resolución del dispositivo o de la orientación de este. El fichero tendrá el mismo nombre en todas las carpetas y el sistema utilizará el que sea necesario en cada momento. Estas carpetas se encuentran en la ruta app\scr\main\res y tenemos tres carpetas.

- Layout que tendrá los ficheros que queramos que aparezcan cuando el terminal está en posición vertical o cuando no hallamos definido un fichero similar en otra de las carpetas.
- Layout-land que almacena los ficheros que serán la definición de las pantallas en el caso de que rotemos nuestro dispositivo y lo pongamos en horizontal.
- Layout-xlarge que tendrá los ficheros que se usarán en aquellos terminales que tengan una resolución alta.

De esta forma, podemos definir distintas formas de visualizar la información repartiendo los controles en la pantalla según nos convenga. En nuestro caso, se ha creado una versión de la pantalla inicial para la orientación vertical y otra para la horizontal. De modo, que en todo momento aprovechamos mejor el espacio de pantalla que tenemos. Por eso si nuestro dispositivo está en horizontal presentaremos los botones en paralelo, mientras que si lo ponemos en posición vertical, aparecerá uno encima de otro.

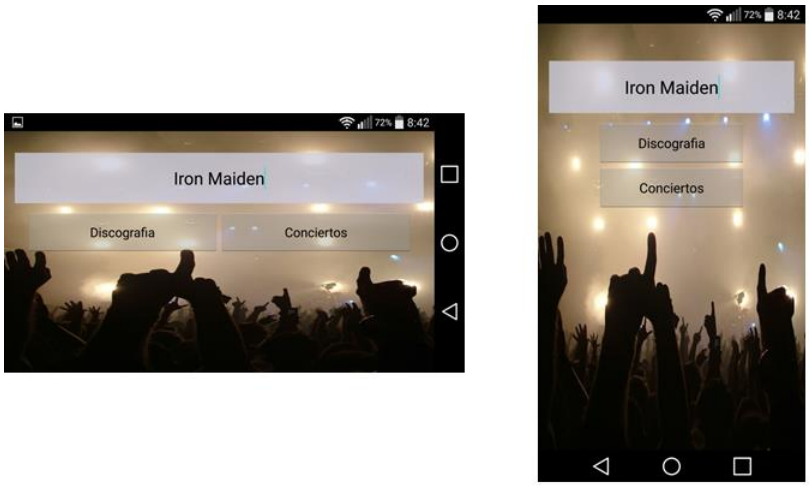

Figura 4-3 Pantalla de inicio en horizontal y vertical

En la figura 4-3., se nos muestra un objeto EditText que nos servirá para introducir el nombre del artista que deseamos buscar, y dos controles Button, uno para la obtención de la discografía y otro para los conciertos. Para que los botones aparezcan en vertical u horizontal se utilizara un LinearLayout con android:orientation="vertical"> o android:orientation=" horizontal" según corresponda

La sentencia en la que se definirá que nuestra actividad tendrá como definición de pantalla dicho fichero xml se realiza mediante la siguiente sentencia.

```
setContentView(R.layout.activity_main);
```

Para seguir explicando el funcionamiento de nuestra aplicación tomaremos como ejemplo el proceso de obtención de la discografía del artista que el usuario desea buscar. La búsqueda de los datos de conciertos, su manejo y visualización para el usuario es similar.

Como ejemplo trabajaremos sacando la discografía de Iron Maiden. Para ello, introduciremos el nombre en el campo de texto y pulsaremos sobre el botón

Discografía. La pulsación del botón desencadenará la llamada al método buscarDisco, cuyo funcionamiento se muestra en la figura 4-4.

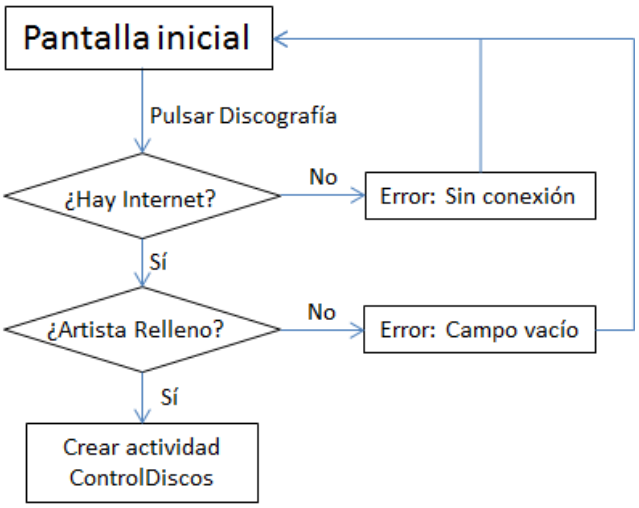

Figura 4-4 Al pulsar botón Discografía. Diagrama de flujo

Y su código es el siguiente:

```
public void buscarDisco(View view) {
	String grupo = group.getText().toString();
	if (hayInternet(this) == false) {
		Toast.makeText(getBaseContext(), getString(R.string.SinConexion),
				Toast.LENGTH_SHORT).show();
		this.finish();
	} else if (grupo.length() == 0) {
		Toast toast1 = Toast.makeText(getApplicationContext(),
				getString(R.string.NoGrupo), Toast.LENGTH_SHORT);
		toast1.show();
	} else {
		// Creamos el Intent
		Intent intent = new Intent(MainActivity.this, ControlDiscos.class);
		// Creamos la información a pasar entre actividades
		Bundle b = new Bundle();
		b.putString("Grupo", grupo);
		intent.putExtras(b);
		// Iniciamos la nueva actividad
		startActivity(intent);}
}
```

Lo primero que haremos es comprobar si nuestro dispositivo tiene acceso a internet mediante la llamada al método hayInternet

```
public static boolean hayInternet(Activity a) {
	boolean hasConnectedWifi = false;
	boolean hasConnectedMobile = false;
	ConnectivityManager cm = (ConnectivityManager) a
				.getSystemService(Context.CONNECTIVITY_SERVICE);
	NetworkInfo[] netInfo = cm.getAllNetworkInfo();
	for (NetworkInfo ni : netInfo) {
		if (ni.getTypeName().equalsIgnoreCase("wifi"))
			if (ni.isConnected())
				hasConnectedWifi = true;
			if (ni.getTypeName().equalsIgnoreCase("mobile"))
			if (ni.isConnected())
				hasConnectedMobile = true;
	}
```

```
return hasConnectedWifi || hasConnectedMobile;}
```

En el caso de que nuestro dispositivo no pueda acceder a internet se mostrará un mensaje de error. Para mostrar mensajes flotantes en Android se hace mediante la clase toast. En nuestro caso sería mediante la siguiente sentencia

```
Toast.makeText(getBaseContext(), getString(R.string.SinConexion),Toast.LENGTH_SHORT).show();
```

En el caso de que tengamos acceso y que el campo de texto esté relleno crearemos un Intent (hilo) para llamar a la siguiente actividad que en nuestro caso sería la clase ControlDiscos.class pasándole como parámetro el nombre del artista.

```
// Creamos el Intent
Intent intent = new Intent(MainActivity.this, ControlDiscos.class);
// Creamos la información a pasar entre actividades
Bundle b = new Bundle();
b.putString("Grupo", grupo);
// Añadimos la información al intent
intent.putExtras(b);
// Iniciamos la nueva actividad
startActivity(intent);
```

En la figura 4-5 podemos ver el diagrama de flujo correspondiente a la búsqueda de la discografía para su visualización por pantalla. A continuación describiremos detalladamente cómo se realiza el proceso.

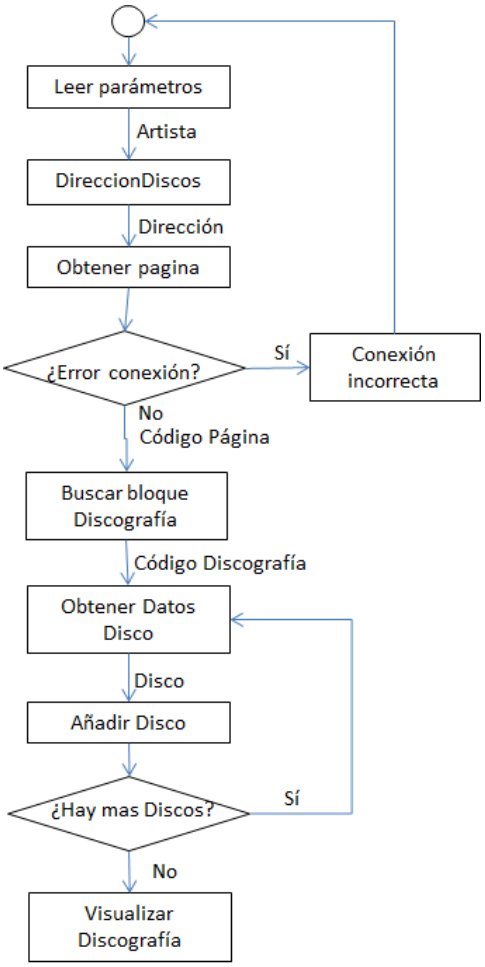

Figura 4-5 Buscar Discografía. Diagrama de flujo

Cuando se desencadena la actividad ControlDiscos, lo primero que hará será recuperar el nombre del grupo que se le ha pasado como parámetro desde la actividad MainActivity que es la que le llamó y obtendremos su valor.

```
Bundle bundle = this.getIntent().getExtras();
grupo = bundle.getString("Grupo");
```

Y con el valor obtenido en la variable grupo procederemos a llamar al método Búsqueda de la clase PagCoveralia, cuyo objetivo es rellenar un ArrayList de la clase Disco con los datos de la página web que contiene la discografía del grupo.

```
public ArrayList<Disco> Busqueda(String grupo) {
	String dir, datos;
	int sw;
	ArrayList<Disco> discos = new ArrayList<Disco>();
	Disco disco = new Disco();
	try {
		grupo = grupo.trim().replaceAll(" +", " ");
		dir = DireccionDiscos(grupo);
		datos = ObtenerPagina(dir, "ISO-8859-1");
		posicion = 0;
		datos = BuscarBloque(datos, "<!--CENTRO-->", "<!--FIN CENTRO-->", 0);
		sw = 1;
		while (sw == 1) {
			disco = DatosDisco(datos, grupo);
			if (disco.getNombre() != "") {
				discos.add(disco);
			} else {
				sw = 0;
			}
		}
	} catch (Exception e) {
		e.printStackTrace();
		return null;
	}
	return discos;
}
```

El primer paso que realizará el método será componer la URL que queremos obtener a partir del artista introducido por el usuario mediante el método DireccionDiscos de nuestra clase PagCoveralia.

```
final String server = "http://www.coveralia.com/";
final String letras = "discografias/";
final String letrapag = "http://www.coveralia.com/letras/";
final String php = ".php";

public String DireccionDiscos(String grupo) {
	String dir;
	grupo = grupo.trim().replaceAll(" +", "-");
	grupo = grupo.trim().replaceAll("ñ+", "n");
	dir = server + letras + grupo.toLowerCase() + php;
	return dir;
}
```

Con el valor de la variable dir obtenida llamaremos al método ObtenerPagina de la clase página que nos devolverá en una variable de tipo String el código HTML de la URL que le pasamos como parámetro. En este método, utiliza la clase

HttpURLConnection para abrir la conexión con internet para que nos devuelva el código de la página.

```
public String ObtenerPagina(String dir, String encoding) {
        String codigo = "";
        final Reader reader;
        final StringBuffer sb = new StringBuffer();
        try {
                URL url = new URL(dir);
                HttpURLConnection conex = (HttpURLConnection) url.openConnection();
                conex.setUseCaches(false);
                if (conex.getResponseCode() == HttpURLConnection.HTTP_OK) {
                        final InputStream is = conex.getInputStream();
                        if (encoding.length() == 0) {
                                reader = new InputStreamReader(is);
                        } else {
                                reader = new InputStreamReader(is, encoding);}
                        final char[] buf = new char[1024];
                        int read;
                        while ((read = reader.read(buf)) > 0) {
                                sb.append(buf, 0, read);
                        }
                        reader.close();
                } else {
                        codigo = ("ERROR: " + conex.getResponseMessage() + "\n");               }
                conex.disconnect();
        } catch (MalformedURLException e) {
                System.out.println("Error de formato");
                return (null);
        } catch (UnknownHostException e) {
                System.out.println("El host no existe o no responde");
                return (null);
        } catch (IOException e) {
                System.out.println("Error desconocido");
                return (null);}
        return sb.toString();
}
```

La página web que nos devolverá a partir de la URL introducida es la que se vemos en la figura 4-6 y sobre la cual mostraremos cómo hemos ido obteniendo la información que usaremos en nuestra aplicación.

Figura 4-6 Discografía de Iron Maiden en Coveralia

En esta página podríamos utilizar la Extensión Web Scraper de nuestro navegador tal y como explicaba en el Capítulo 2 y podríamos obtener de una forma rápida y sencilla la lista de los publicados por Iron Maiden y que están catalogados en esta página [figura4-7], y de esta forma podríamos tener una lista de los discos para pegarla por ejemplo en un documento de texto.

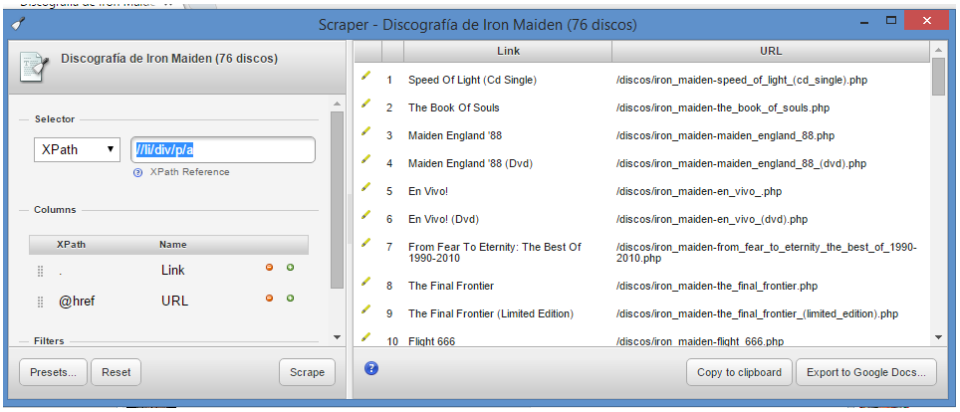

Figura 4-7 Discografía de Iron Maiden en Web Scraper.

Pero nosotros necesitamos un análisis más detallado de la estructura de la página puesto no que solo queremos conocer el nombre de los discos sino otros datos como el año o la url que nos llevará a la lista de canciones.

Para poder analizar con más detalle la estructura de la página que deseamos tratar, usaremos la opción Inspeccionar Elemento de nuestro navegador. Para ello, debemos pulsar el botón derecho sobre la página en la parte que aparecen los discos y como resultado obtendremos el código HTML del elemento seleccionado, tal y cómo vemos en la figura 4-8.

Figura 4-8 Inspeccionar Elemento Bloque discografía

En el código fuente de la página buscaremos con las etiquetas "<!--CENTRO-->", "<!--FIN CENTRO-->" para obtener la parte correspondiente al HTML donde aparece la información referente a cada uno de los discos del artista que estamos buscando. Para obtener bloques de código dentro del String que contiene toda la página lo haremos mediante la llamada al método BuscarBloque de la clase página que nos devolverá un String con el HTML existente entre eti_ini y eti_fin,

```
public String BuscarBloque(String codigo, String eti_ini, String eti_fin, int pos) {
        int ant = 0;
        String bloque = "";
        ant = codigo.indexOf(eti_ini, pos);
        if (ant >= 0) {
                ant = ant + eti_ini.length();
                if (eti_fin != "") {
                        pos = codigo.indexOf(eti_fin, ant);
```

```
                } else {
                        pos = codigo.length();
                }
                bloque = codigo.substring(ant, pos);
        }
        posi = pos;
        return bloque;
}
```

Ya que disponemos de la parte de código que nos interesa en la que aparece la información de los discos, mediante Inspeccionar elemento observamos que la información de cada uno de los discos viene delimitada por las etiquetas <li> y cómo podemos ver en la figura 4-9.

Figura 4-9 Inspeccionar elemento Lista de discos

Ahora, que ya sabemos cómo está estructurado el listado con toda la discografía, el siguiente paso será analizar el código entre ambas etiquetas para saber cómo está estructurado dicho código HTML que nos muestra la información correspondiente a cada uno de los discos

```
<li>
<div class="imagen"><a href="/discos/iron_maiden-the_book_of_souls.php" title="Disco The Book Of
Souls de Iron Maiden"><img src="http://images.coveralia.com/audio/thumbs/233371.jpg" alt="Disco The
Book Of Souls de Iron Maiden" class="i50" /></a></div>
<div class="info">
<p class="usuario"><a href="/discos/iron_maiden-the_book_of_souls.php">The Book Of Souls</a></p>
<p>2015, Warner</p>
</div>
<div class="clear"></div>
</li>
```

A partir de este código HTML en el que viene la información referente a cada disco, obtendremos mediante la búsqueda por etiquetas de inicio y fin de cadena los datos de cada uno de los atributos que deseamos almacenar para la clase Disco. Por ejemplo, para el atributo urlInfo, utilizaremos como etiqueta de inicio *<a href="* y *"* como etiqueta de fin. De esta misma forma y de manera secuencial iremos buscando el resto de los datos que aparecen en este trozo de código y son necesarios para rellenar los atributos de nuestra clase disco.

Este proceso lo realizamos mediante la clase DatosDisco que a partir del trozo de código perteneciente al disco, nos devuelve un objeto de la clase Disco con todos sus atributos rellenos.

```
public Disco DatosDisco(String codigo, String grupo) {
        String bloque = "", url = "", trozo = "";
        Disco disco = new Disco();
        int sw;
        pos_bloq = 0;
```

```
pos_url = 0;
bloque = BuscarBloque(codigo, "<li>", "</li>",posicion);
posicion = posi;
trozo = BuscarBloque(bloque, "<img src=\"", "\"", pos_bloq);
disco.setPortada(trozo);
trozo = BuscarBloque(bloque, "<p class=\"usuario\">", "</p>", pos_bloq);
pos_bloq = posi;
url = BuscarBloque(trozo, "<a href=\"/", "\"", 0);
url = server + url;
disco.setUrlInfo(url);
disco.setNombre(LimpiaCadena(trozo));
trozo = BuscarBloque(bloque, "<p>", "</p>", pos_bloq);
disco.setAnyo(trozo);
pos_bloq = posi;
disco.setGrupo(grupo);
sw = 1;
return disco;}
```

En este caso, los datos obtenidos a partir de este trozo de código que aparecía en la parte de arriba serían los siguientes.

nombre: The Book Of Souls
grupo: Iron Maiden
anyo: 2015, Warner
urlInfo: /discos/iron_maiden-the_book_of_souls.php
portada: http://images.coveralia.com/audio/thumbs/233371.jpg

Este proceso lo realizamos mientras en el código de la página HTML sigan apareciendo discos, los cuales los iremos añadiendo como elementos de nuestra lista para su posterior visualización.

Una vez que tenemos nuestra lista de discos cargada, el siguiente paso será mostrársela al usuario por pantalla para que pueda moverse por ella y consultar la información. Para visualizar la información, lo haremos mediante el layout ver_discos.xml.

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
  <ListView android:id="@android:id/list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
  <TextView android:id="@android:id/empty"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
  </LinearLayout>
```

Este layout no es más que una lista en la que se mostrarán objetos del tipo control_discos.xml que es un control personalizado que hemos creado para mostrar la información de cada disco.

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout xmlns:android=http://schemas.android.com/apk/res/android
        android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:id="@+id/lineItem">
        <LinearLayout android:gravity="center_vertical"
                android:layout_height="wrap_content" android:layout_width="match_parent"
```

```
                android:orientation="horizontal" android:padding="5dp">
                <ImageView android:layout_height="70dp"
                        android:layout_width="70dp" android:id="@+id/Portada" />
                <LinearLayout android:orientation="vertical" android:layout_width="match_parent"
                        android:layout_height="match_parent" android:layout_marginLeft="5dp"
                        android:gravity="center_vertical">
                        <TextView android:layout_height="wrap_content"
                                android:layout_width="fill_parent" android:id="@+id/Disco"
                                android:scrollHorizontally="true" android:ellipsize="end"
                                android:textSize="18sp" android:maxLines="1" />
                        <TextView android:maxLines="1"  android:layout_height="wrap_content"
                                android:layout_width="fill_parent" android:id="@+id/Anyo"
                                android:scrollHorizontally="true" android:ellipsize="end"
                                android:textSize="16sp" />
                 </LinearLayout>
        </LinearLayout>
</RelativeLayout>
```

En este control mostramos la imagen correspondiente a la portada así como el nombre del disco y su año de edición y discográfica. Para su diseño buscamos distintos ejemplos en internet en los que aparecían la utilización de los distintos controles usados así como las propiedades que debía usar para obtener el resultado deseado. Sobre esta pantalla el usuario tendrá la posibilidad de pulsar sobre la imagen para ver la portada en grande o sobre el nombre del disco para obtener la lista de canciones que lo componen.

Para el relleno de la lista, lo primero que haremos es crearnos una clase contenedor que tendrá las variables correspondientes a los datos que vamos a mostrar más el número de línea que posteriormente utilizaremos para detectar qué línea hemos pulsado

```
class ContenedorDisco {
        TextView nombre;
        TextView anyo;
        ImageView Portada;
        RelativeLayout row;
}
```

Al objeto de esta clase contenedor, lo primero que haremos es asignarle el elemento correspondiente y posteriormente el valor que tendrá cada uno de esos objetos para su visualización por pantalla.

```
public View getView(final int arg0, View view, ViewGroup parent) {
        ContenedorDisco contenedor;
        Portada portada = new Portada();
        Bitmap imagen;

        if (view == null) {
                view = lInflater.inflate(R.layout.control_discos, null);
                contenedor = new ContenedorDisco();
                contenedor.nombre = (TextView) view.findViewById(R.id.Disco);
                contenedor.anyo = (TextView) view.findViewById(R.id.Anyo);
                contenedor.Portada = (ImageView) view.findViewById(R.id.Portada);
                contenedor.row = (RelativeLayout) view.findViewById(R.id.lineItem);
                view.setTag(contenedor);
                } else
                        contenedor = (ContenedorDisco) view.getTag();
                Disco discos = (Disco) getItem(arg0);
                contenedor.nombre.setText(discos.getNombre());
                contenedor.anyo.setText(discos.getAnyo());
                imagen = portada.downloadBitmap(discos.getPortada());
                contenedor.Portada.setImageBitmap(imagen);
```

Ahora que ya le hemos mostrado al usuario la lista con los discos del artista seleccionado, ya tendrá la posibilidad de interactuar con la aplicación. Desde aquí el usuario podrá pulsar sobre la portada lo que hará que esta se muestre a pantalla completa. Para ello, tendremos que detectar el método OnClick del objeto Portada de nuestra línea contenedor. Mediante el método getItem() obtenemos cuál es la línea sobre la que ha pulsado el usuario de la que obtendremos su clase disco, y llamaremos a la actividad ControlPortda pasándole el disco correspondiente a la línea seleccionada para que se muestre la imagen a pantalla completa.

```
contenedor.Portada.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
            ArrayList<Caratula> caratulas = new arrayList<Caratula>();
            Disco discoselec = (Disco) getItem(arg0);
            caratulas = pagina.ObtenerCaratulas(discoselec.getUrlInfo());
            discoselec.caratulas.addAll(caratulas);
            // Creamos el Intent
            Intent intent = new Intent(ControlDiscos.this, ControlPortada.class);
            Bundle b = new Bundle();
            intent.putExtras(b);
            intent.putExtra("Disco", discoselec);
                    startActivity(intent);
        }
});
```

Cuando la clase Control Portada es llamada lo primero que hace es obtener la url que corresponde a la portada del disco que recibe desde la pantalla anterior. Con esta url, procede a realizar la llamada al método downloadBitmap de la clase portada que nos devolverá el objeto imagen de la clase Bitmap para mostrarla posteriormente con la llamada al método setImageBitmap de nuestro objeto ImageView que tenemos definido en el layout portada.

```
Disco disco = bundle.getParcelable("Disco");
url = disco.caratulas.get(0).urlHigh;

imagen = portada.downloadBitmap(url);
iv.setImageBitmap(imagen);
```

El método downloadBitmap de la clase portada lo que hace es abrir una conexión HTTP para obtener el código de la url que contiene la imagen de la portada el cual convertirá en un objeto Bitmap mediante el método BitmapFactory.decodeStream para que pueda ser visualizado por Android.

```
public Bitmap downloadBitmap(String url) {
        Bitmap image = null;
        final DefaultHttpClient client = new DefaultHttpClient();
        // forming a HttoGet request
        final HttpGet getRequest = new HttpGet(url);
        try {
                HttpResponse response = client.execute(getRequest);
                final HttpEntity entity = response.getEntity();
                inputStream = entity.getContent();
                image = BitmapFactory.decodeStream(inputStream);
                return image;}
        }
```

La otra opción que se le presenta al usuario en la lista de discos, es pulsar directamente sobre la línea lo que desencadenará el evento onItemClick. Este método recuperará el disco seleccionado y mediante el método ObtenerCanciones de la clase

PagCoveralia obtendrá la lista de las canciones que componen el disco que hemos seleccionado las cuales almacenaremos en el atributo canciones de nuestra clase Disco.

Ya que tenemos las canciones, procederemos a crear una nueva actividad de la clase ControlCanciones que nos mostrará los datos del disco así como la lista de las canciones que lo componen.

```
public void onItemClick(AdapterView<?> parent, android.view.View v,
        int position, long id) {
        Disco discoselec = ((Disco) parent.getAdapter().getItem(position));
        Intent intent = new Intent(ControlDiscos.this, ControlCanciones.class);
        Bundle b = new Bundle();
        b.putInt("Opcion", 2);
        intent.putExtras(b);
        intent.putExtra("Disco", discoselec);
        startActivity(intent);
        }
});
```

La clase ControlCanciones que sirve para mostrar una lista de canciones es usada tanto para mostrar las canciones que forman parte del disco seleccionado, como para visualizar las canciones que interpretó el artista en el concierto que deseábamos consultar. La pantalla se compone en la parte superior de una serie de objetos TextView en los que se visualiza los datos del disco o del concierto, y en la parte inferior la lista de canciones que forman parte del disco o concierto. La información de la parte superior de la pantalla siempre quedará fija para que pueda ser consultada aunque nos desplazamos con el scroll en la lista de canciones.

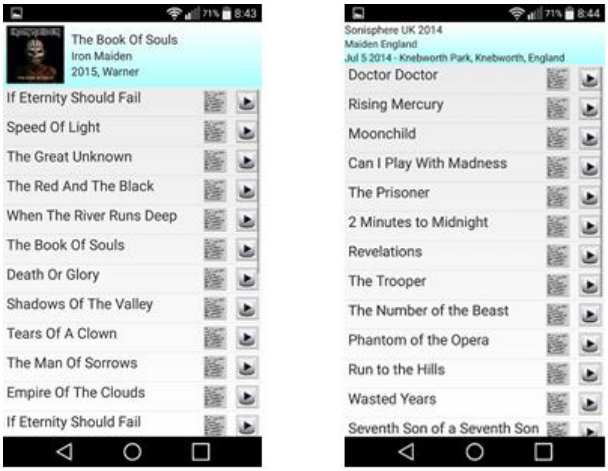

Figura 4-10 Pantalla Lista canciones disco y concierto

Como he comentado la parte superior en la que aparece la información del disco o el concierto es distinta en función de lo que estemos mostrando en la lista. Esto lo podemos ver la figura4-10 donde se muestran los dos casos. Para ello, nos hemos creado dos layout distintos los cuales serán llamados desde la clase ControlCanciones en función de si ha sido llamada desde la pantalla con la discografía o desde la lista de los conciertos.

```
switch (opcion) {
        case 1:
                setContentView(R.layout.ver_canciones_con);
        case 2:
                setContentView(R.layout.ver_canciones);
        }
```

En esta pantalla, se le muestra al usuario dos botones con los que podrá interactuar. Con el primer botón, podrá visualizar la letra de la canción seleccionada la cual será obtenida de la página de coveralia, y pulsando sobre el segundo se puede reproducir el video de YouTube de la canción seleccionada a la vez que se visualiza la letra.

Cada uno de estos botones, tiene una imagen que identifica de forma instintiva la operación que realiza. Estas imágenes que buscamos en internet para encontrar la que mejor se adaptaba a lo que queríamos que apareciera en el botón debían ser almacenadas en nuestro proyecto. Para ello, como comentamos con anterioridad cuando creamos el proyecto Android se nos crearon una serie de carpetas sobre las que trabajar. En esta caso, para el almacenamiento de las imágenes. Dentro de la ruta app\scr\main\res tenemos las carpetas drawable-hdpi, drawable-mdpi, drawable-xhdpi y drawable-xxhdpi en las que almacenaremos las imágenes que aparecen en la aplicación en función del tamaño que estas tendrán.

Dentro de estas carpetas, también podemos definir los archivos XML en los que nos definiremos colores que podremos usar posteriormente en nuestros layouts. En nuestro caso, definiremos un nuevo color que utilizaremos para que se diferencie mejor la parte superior con la información del disco o concierto de la lista de canciones. Para ello, nos hemos creado el archivo color.xml en el que queremos obtener un color que se irá degradando hasta convertirse en otro. Para ello, le debemos definir el color inicial y el final y la forma en la que un color se convertirá en el otro.

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="rectangle">
  <gradient
     android:type="linear"
     android:startColor="#FFFFFFFF"
     android:endColor="#FFAAffff"
     android:angle="270" />
</shape>
```

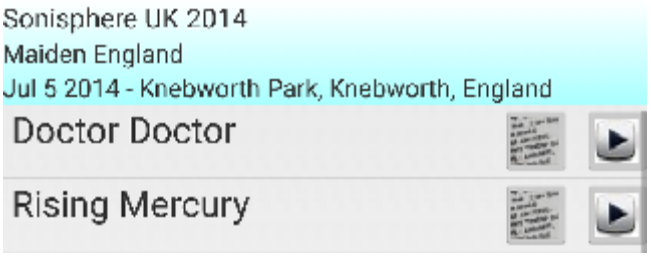

Figura 4-11 Uso de los ficheros de color

Este color posteriormente lo utilizaremos en la definición del layout poniéndoselo como color de fondo al objeto y obtendremos el resultado que vemos en la figura 4-11.

```
android:background="@drawable/color"
```

Después de la explicación de la inclusión de los ficheros de imágenes y colores en las carpetas drawable, continuamos con la explicación de las opciones que ofrece el ControlCanciones. Como hemos comentado desde esta pantalla podremos leer la letra de la canción o ver el video en YouTube además de poder estar leyendo la letra. Como la parte correspondiente a la visualización de la letra de la canción se encuentra en las dos opciones, procederemos únicamente a explicar la opción Play Video.

Pero antes de pasar a describir la forma en la que definitivamente se ha obtenido el dato del video que queremos reproducir y la forma de reproducirlo en nuestra aplicación, pasamos a describir el proceso que ha tenido esta parte del proyecto a lo largo del tiempo porque ha sufrido varios cambios que demuestran cómo ha avanzado la obtención de información de la red mediantes distintas herramientas.

Lo primero que intentamos fue realizar la búsqueda de Iron maiden desde el navegador y vimos que el resultado obtenido era

https://www.youtube.com/results?search_query=iron+maiden+speed+of+light

por lo cual, como hacíamos con el resto de las páginas a las que accedíamos, lo primero era componer mediante programa dicha dirección para obtener la página y poder buscar en ella los datos deseados. Pero aquí, nos encontramos que la página que devolvía dicha URL no tenía el contenido que esperábamos puesto que el código fuente lo único que nos traía era un error informando que no se podía obtener la página solicitada.

Ante este problema, comenzamos a buscar información en Internet para ver cómo era posible obtener una lista de videos de Youtube desde una aplicación Android y encontramos la existencia de una API proporcionada por YouTube que nos permitía interactuar con dicha página. En este caso, se trataba de la versión 2.0. Para poder obtener la consulta deseada se tenía que realizar la llamada componiendo la dirección tal y como se muestra a continuación.

https://gdata.youtube.com/feeds/api/videos?q=iron+maiden+speed+of+light &
orderby=relevance&format=6

En esta dirección además de indicarle en los parámetro los campos por los que queríamos buscar, le solicitábamos los videos ordenados por relevancia (orderby=relevance) y que únicamente nos devolviera aquellos videos que se pudieran reproducir en un dispositivo móvil.

Desde nuestra aplicación pasándole esta URL ya sí obteníamos un código fuente en el que se mostraban los videos que correspondían a nuestra consulta ordenamos por relevancia y realizando una búsqueda por bloques al igual que se realizaba para el resto de las páginas ya podemos obtener la dirección correspondiente para poder reproducir el video. En este caso la dirección tenía el siguiente formato.

https://www.youtube.com/watch?v=X4bgXH3sJ2Q

Con dicha dirección, llamábamos a un intent del tipo Internet pasándole dicha dirección. Al recibir esta solicitud, el sistema Android reconocía que la dirección que le estábamos pasando correspondía a YouTube y nos mostraba una pantalla en la que nos permitía ejecutar la acción directamente en YouTube o en cualquiera de los navegadores que teníamos instalado en mi terminal.

```
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(urlYT));
```

Después de varios meses probando la aplicación en el smartphone un día al ir a visualizar un video la aplicación daba error. Al comenzar a investigar que podía estar pasando revisando foros en internet descubrimos que el problema estaba en que Google había dejado de tener operativo el API de Youtube 2.0 y había comenzado a funcionar la versión 3.0 de dicha API. Dicho cambio supuso realizar varios cambios en la aplicación que a la larga fueron beneficiosos para el resultado final.

Lo primero que había que hacer para poder realizar llamadas a la nueva API era dar de alta el proyecto en la Google Developers Console y habilitar el uso de la API YouTubeData para dicha aplicación lo cual nos permitiría llamar a la API desde nuestra aplicación. *[developers]*

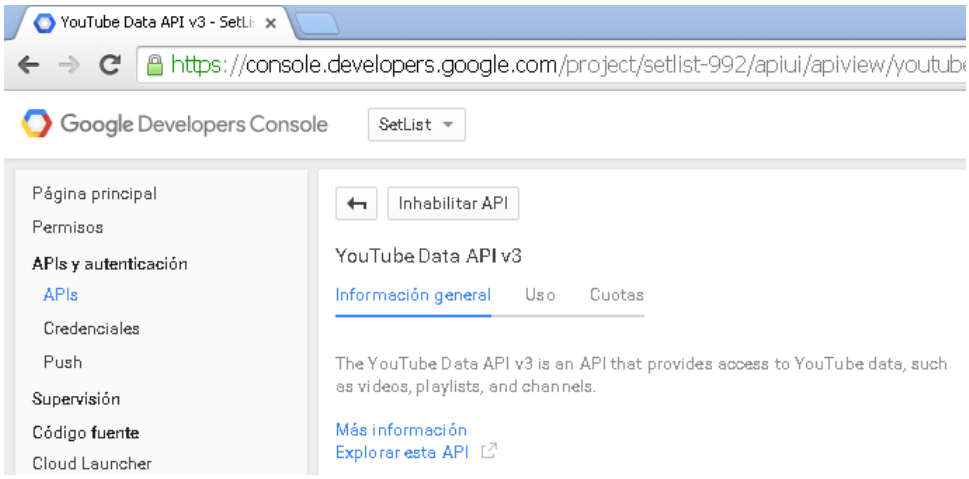

Figura 4-12 Habilitar API YouTube Data v3

Una vez que estaba habilitada ela API [Figura 4-12], el siguiente paso era obtener la credencial para aplicaciones Android la cual debíamos usar a la hora de realizar llamadas a la API desde nuestra aplicación.

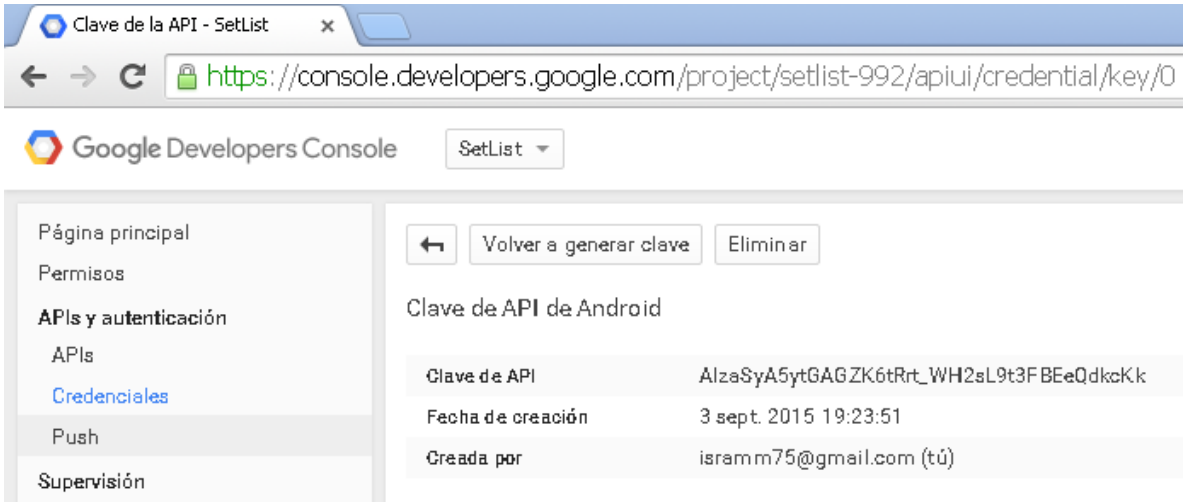

Figura 4-13 Creación de credencial

Ya que teníamos la clave con la que se debía llamar a la API [Figura 4-13], el siguiente paso fue modificar la aplicación para permitirle al usuario reproducir el video de la canción deseada. En este punto, se nos presentaban dos opciones a la hora de realizar la consulta para obtener la lista de videos de la canción seleccionada.

La primera opción era componer la URL para la busqueda segun la guía de la API, lo cual nos daría un fichero JSON (*JavaScript Object Notation*) que tendríamos que tratar para obtener el ID del primer video que aparecía en la lista que es el que deseamos reproducir. En este caso, la dirección de búsqueda quedaría de la siguiente manera.

https://www.googleapis.com/youtube/v3/search?part=snippet&q=iron+maiden+speed+of+light&order=relevance&key=AIzaSyA5ytGAGZK6tRrt_WH2sL9t3FBEeQdkcKk

El fichero JSON se trataría mediante las librerías que nos proporciona Android. Es este fichero, se nos proporciona bajo la etiqueta items la información de cada uno de los videos que cumplen con los parámetros de búsqueda que le hemos pasado ordenados por relevancia. De aquí tendríamos que sacar el valor del parámetro videoid para poder reproducir el video deseado.


```
{
 "kind": "youtube#searchListResponse",
 "etag": "\"jOXstHOM20qemPbHbyzf7ztZ7rI/GLMtkSTBjB2QdMWGBaerwbDa2ys\"",
 "nextPageToken": "CAUQAA",
 "pageInfo": {
  "totalResults": 67715,
  "resultsPerPage": 5
 },
 "items": [
  {
   "kind": "youtube#searchResult",
   "etag": "\"jOXstHOM20qemPbHbyzf7ztZ7rI/EUKaDPLxyB6QF3SPJ9cGr0dXCUo\"",
   "id": {"kind": "youtube#video",
    "videoId": "-F7A24f6gNc"
   },
   "snippet": {
    "publishedAt": "2015-08-14T07:00:12.000Z",
    "channelId": "UCaisXKBdNOYqGr2qOXCLchQ",
    "title": "Iron Maiden - Speed Of Light (Official Video)",
    "description": "Iron Maiden - Speed Of Light - From the new album 'The Book Of Souls' - out now. Play
the Speed Of Light Game: http://speedoflight.ironmaiden.com Get the ...",
    "thumbnails": {
     "default": {"url": "https://i.ytimg.com/vi/-F7A24f6gNc/default.jpg"},
     "medium": {"url": "https://i.ytimg.com/vi/-F7A24f6gNc/mqdefault.jpg"},
     "high": {"url": "https://i.ytimg.com/vi/-F7A24f6gNc/hqdefault.jpg"}
    },
    "channelTitle": "ironmaiden",
    "liveBroadcastContent": "none"
   }
  },
```


La segunda opción que teníamos y que es por la que hemos optado porque era la más fácil de trabajar, consiste en utilizar la clase correspondiente al API de YouTube Data V3. Esta clase nos proporciona muchas de las funciones que se pueden realizar en YouTube desde el navegador, tales como publicar videos, realizar búsquedas por contenidos de videos o listas de reproducción o incluso borrarlos. En nuestro caso, la parte que usaremos será el método search que nos proporcionará la lista de videos que se corresponde con la canción seleccionada por el usuario desde la aplicación.

Para realizar la búsqueda nos hemos creado el objeto query que es un objeto de la clase YouTube.Search.List y lo hemos inicializado rellenado a cada uno de los parámetros que servirán para obtener la lista de videos que deseamos.

Lo primero es pesarla nuestra clave o credencial que nos ha proporcionado Google al habilitar la Api en la Google Developers Console y que servirá para que nuestra aplicación se identifique a la hora de realizar la búsqueda. Esta identificación servirá a Google para controlar el acceso y el número de peticiones el cual nosotros también podremos consultar desde la consola.

El siguiente parámetro que utilizamos será el setType para indicarle que queremos tratar un video, puesto que la API nos permite trabajar con videos, *playlist* o canales. Posteriormente, mediante el parámetro setFields le indicamos la lista de campos que queremos que nos devuelva la consulta.

```
public YoutubeConnector(Context context) {
   youtube = new YouTube.Builder(new NetHttpTransport(),
         new JacksonFactory(), new HttpRequestInitializer() {
      @Override
      public void initialize(HttpRequest hr) throws IOException {}
   }).setApplicationName("setlist").build();
   try{
      query = youtube.search().list("id,snippet");
      query.setKey(KEY);
      query.setType("video");
      //query.setOrder('');
      query.setFields("items(id/videoId,snippet/title,snippet/description,snippet/thumbnails/default/url)");
   }catch(IOException e){
      Log.d("YC", "Could not initialize: " + e);
   }
}
```

Una vez que está inicializado el objeto query el siguiente paso será llamar al método SetQ para pasarle la lista de palabras que componen la búsqueda que deseamos realizar separadas por el signo +. En nuestro caso, el nombre del grupo y la canción seleccionada iron+maiden+speed+of+light. Ahora ya solo nos queda ejecutar la consulta y tratar los datos que nos devuelva la lista.

Para almacenar la lista de videos que nos devuelve la búsqueda lo hacemos sobre una lista de la clase VideoItem que nos hemos creado con anterioridad y la cual tendrá como propiedades cada uno de los campos que habíamos pasado a la query en el parámetro SetFields.

```
public List<VideoItem> search(String keywords){
   query.setQ(keywords);
   try{
      SearchListResponse response = query.execute();
      List<SearchResult> results = response.getItems();
      List<VideoItem> items = new ArrayList<VideoItem>();
      for(SearchResult result:results){
         VideoItem item = new VideoItem();
         item.setTitle(result.getSnippet().getTitle());
         item.setDescription(result.getSnippet().getDescription());
         item.setThumbnailURL(result.getSnippet().getThumbnails().getDefault().getUrl());
         item.setId(result.getId().getVideoId());
         items.add(item);
      }
      return items;
   }catch(IOException e){
      Log.d("YC", "Could not search: "+e);
      return null;
   }
}
```

Ya que tenemos la lista de videos correspondiente a nuestra búsqueda, el siguiente paso será trabajar con la API YouTubePlayer. Para ello, cogeremos el valor de la propiedad SetId del primer elemento de la lista puesto que esta estará ordenada por relevancia ya que a la hora de definir la query no le hemos indicado nada en la propiedad order y por defecto esta tendrá el valor *relevance*.

La API YouTubePlayer que nos proporciona Google nos permite la funcionalidad de incluir en nuestra aplicación Android un reproductor de videos de YouTube. Con ella, se nos permite reproducir videos y listas de reproducción, pudiendo personalizar el aspecto de reproductor y mediante eventos controlar las operaciones que el usuario realice sobre el reproductor tales como reproducir, pausar o buscar un punto específico del video.

Para poder usar la API tuvimos que bajar las librerías desde la página de Google Developers e incluirlas en el proyecto en otra de las carpetas que se nos crearon al inicio del proyecto. En este caso, se debían incluir en la carpeta libs para que después desde las distintas clases del programa pudiera crear un objeto de la clase YouTubePlayerView para trabajar con ella.

La ventaja que nos ofrece el uso de la API YouTubePlayer sobre la llamada que hacíamos al navegador Web del dispositivo para reproducir el video deseado, es que nos permite controlar el aspecto del reproductor así como su funcionamiento.

Ya que podíamos controlar el aspecto decidimos que a la hora de reproducir el video una buena opción sería reproducir el video y al mismo tiempo proporcionarle al usuario la posibilidad de leer la letra de la canción seleccionada. Para ello, se definió un layout en el que en la parte superior aparecería un control YouTubePlayer y debajo un TexTView con scroll para poder visualizar la letra.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">
  <com.google.android.youtube.player.YouTubePlayerView
        android:id="@+id/youtube_player"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
  <TextView android:id="@+id/Letra"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="20sp" />
  </LinearLayout>
```

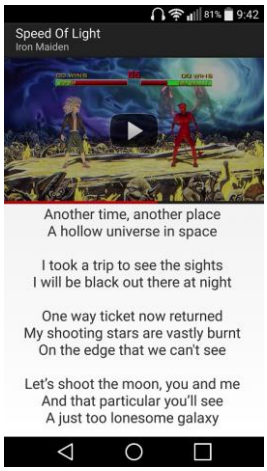

Figura 4-14 Pantalla de Reproducción de video y letra

Ya que tenemos definido el layout ahora habrá que proceder a cargar el reproductor de YouTube y la letra correspondiente a la canción que el usuario desea visualizar, de la forma que vemos en la figura 4-14. Lo primero será obtener los parámetros que se le han pasado al intent y con ellos componer la cadena de búsqueda que le pasaremos al API YouTube Data y obtendremos del valor del Id del primer elemento de la lista devuelta. Con este valor se llama al método LoadVideo del player de Youtube que hará que comience a reproducirse el video.

Para la obtención de la letra, primero debemos componer la URL correspondiente a la canción que deseamos buscar en la página de coveralia y una vez que tenemos la URL, mediante el método obtendremos el código con la letra de la canción. Dicha letra la mostraremos en el objeto TextView el cual tendrá activado el scroll vertical que nos permitirá desplazarnos para visualizar la letra.

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  grupo = bundle.getString("Grupo");
  cancion = bundle.getString("Cancion");
  url = bundle.getString("Url");

  handler = new Handler();

  busqueda = grupo + "+" + cancion;
  searchOnYoutube(busqueda);
  VIDEO_ID = searchResults.get(0).getId();
  setContentView(R.layout.ver_youtube);
  TextView txtLetra = (TextView) findViewById(R.id.Letra);
  /** Initializing YouTube player view **/
  youTubeView = (YouTubePlayerView) findViewById(R.id.youtube_player);
  youTubeView.initialize(API_KEY, this);
  if (url == null){
     url = pagCover.DireccionLetras(cancion, grupo);
  }
  letra = pagCover.ObtenerLetra(url);
  String s = "";
  for(int x=0;x<=100;x++){
     s += "Line: "+String.valueOf(x) + "\n";        }
  txtLetra.setMovementMethod(new ScrollingMovementMethod());
  // Construimos el mensaje a mostrar
  txtLetra.setText(letra);
}
```

```
public void onInitializationSuccess(Provider provider, YouTubePlayer player, boolean wasRestored) {
    /** add listeners to YouTubePlayer instance **/
    /** Start buffering **/
    if (!wasRestored) {
        player.loadVideo(VIDEO_ID);
        // Hiding player controls
        player.setPlayerStyle(YouTubePlayer.PlayerStyle.MINIMAL);}
}
```

Ahora que ya tenemos terminada nuestra aplicación solo tenemos que probar que funciona antes de pasarla a nuestro dispositivo móvil. Para ello, el SDK de Android nos proporciona un emulador para que podamos simular que estamos trabajando con un dispositivo móvil. Incluso nos permite probar con distintas resoluciones, cómo vemos en la figura 4-15, que el sistema tiene cargadas y que corresponden con algunos de los terminales más usados para ver cómo quedaría la disposición en los layout de nuestros objetos en distintos terminales.

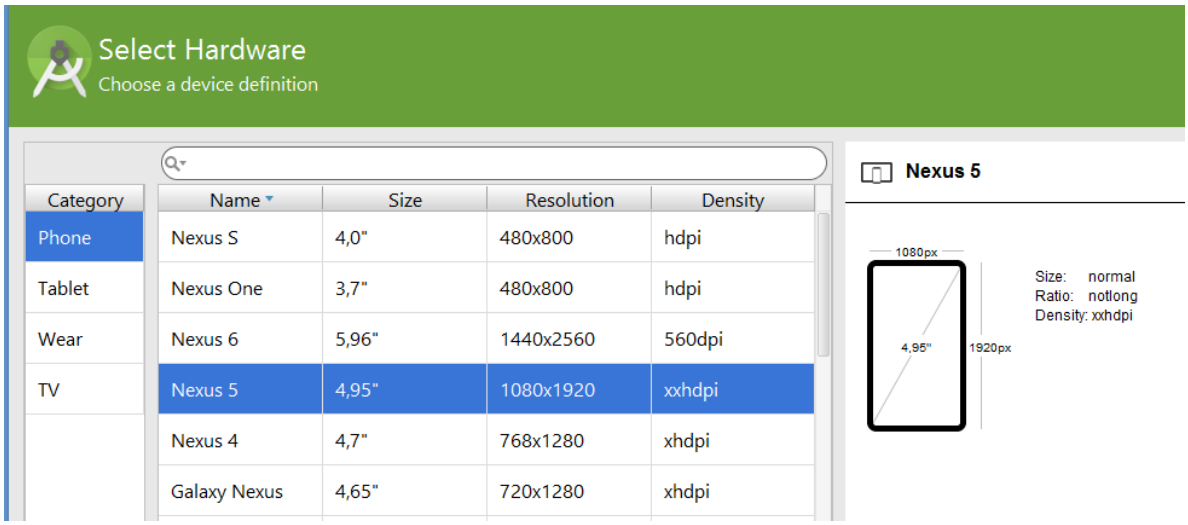

Figura 4-15 Resoluciones de pantalla del emulador

En la figura 4-16 vemos el emulador al lanzar nuestra aplicación para probar que funciona según lo esperado.

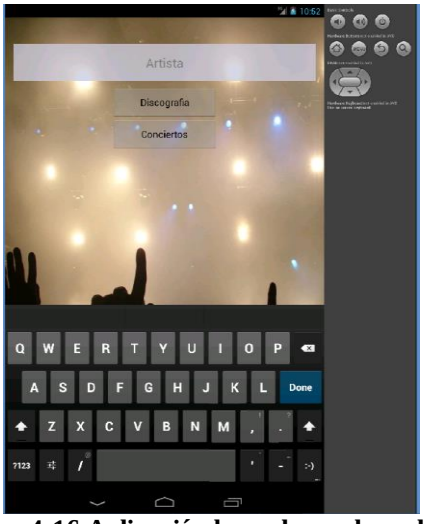

Figura 4-16 Aplicación lanzada en el emulador

4.4 Generación del archivo APK

Una vez que ya hemos comprobado en el emulador que nuestra aplicación funciona de forma correcta y el aspecto que tiene es el que nosotros deseábamos, ya solo nos queda generar el archivo APK para poderla instalar en nuestro dispositivo móvil. Para poder generar el archivo tendremos que cambiar en la ventana de Build variant la opción Debug que era la que teníamos hasta ahora seleccionada para poder lanzar la aplicación en el simulador por Release que es la que nos permitirá compilar la aplicación y construir el archivo APK. *[android01]*

Para generar el fichero deberemos seleccionar la opción Generate Singed APK del del menú Build. Lo primero que nos pedirá serán nuestros datos y la contraseña que deseamos ponerle a la clave que generaremos. [Figura4-17]

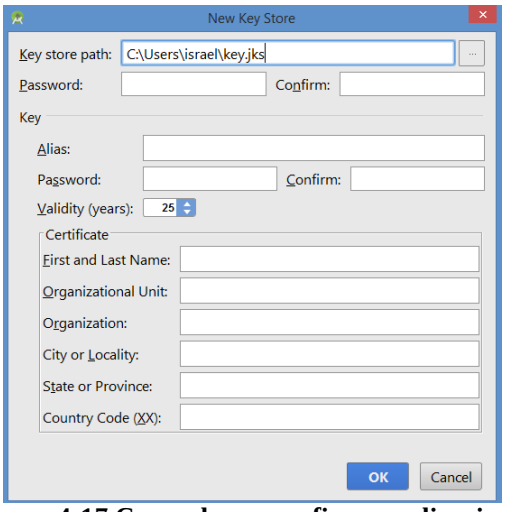

Figura 4-17 Crear clave para firmar aplicaciones

La clave que nos permitirá firma nuestras aplicaciones Android se almacenará en el fichero key.jks. Tenemos que tener mucho cuidado de conservar el fichero con la clave, porque en el caso de que lo perdamos, no podremos firmar nuestras aplicaciones y deberemos generarnos una nueva clave. El siguiente paso es firmar nuestra aplicación con dicha clave indicándole las password que habían puesto en el paso anterior. [Figura 4-18]

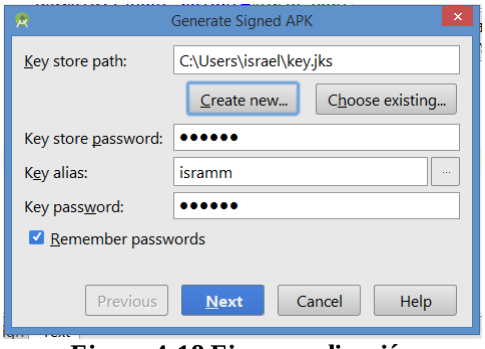

Figura 4-18 Firmar aplicación

El siguiente paso sería indicarle la ruta en la que deseamos dejar almacenado el fichero APK generado. Y si todo ha ido correcto nos mostrara el mensaje que vemos en

la figura 4-19 informándonos que el fichero ha sido creado correctamente en la ruta que habíamos seleccionado.

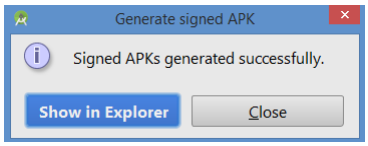

Figura 4-19 Fichero APK creado con éxito.

Ahora que ya tenemos el archivo de instalación generado, solo nos queda instalarlo en nuestro dispositivo móvil para poder comenzar a consultar la discografía y los conciertos de nuestros grupos favoritos. Y por supuesto ver los videos de las canciones a la vez que leemos la letra.

5 Resultado final

En la figura 5-1, se muestra cuáles son las distintas pantallas de las que consta la aplicación que hemos desarrollado y como se relacionan entre sí.

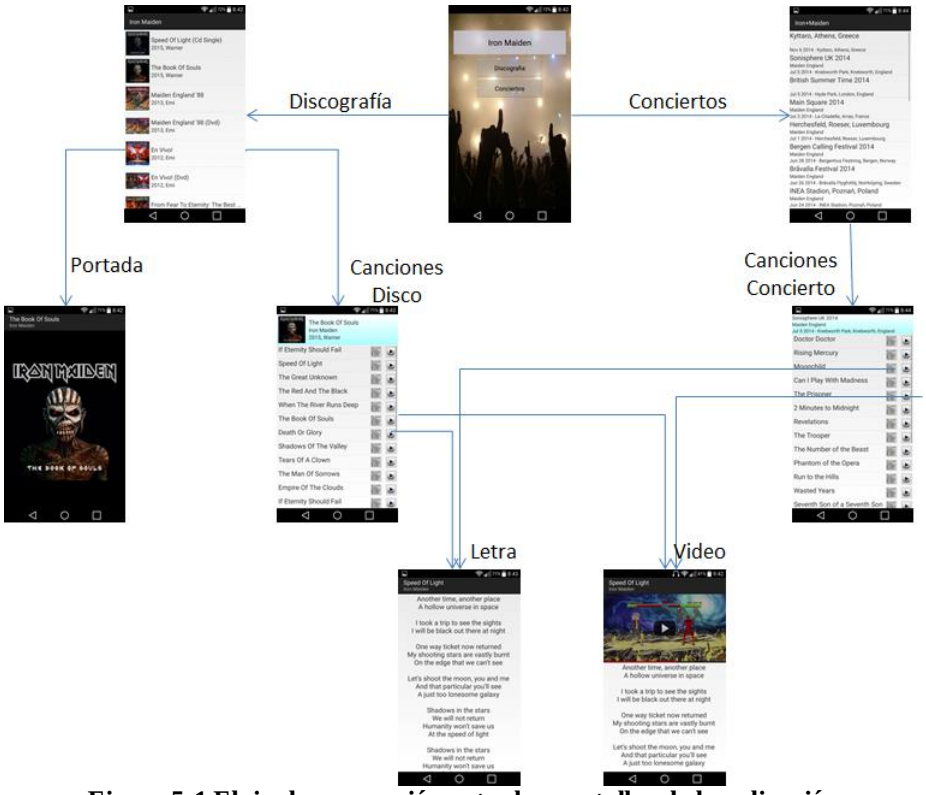

Figura 5-1 Flujo de navegación entre las pantallas de la aplicación

Aunque durante el Capítulo 4, se ido describiendo cómo se pasaba de una pantalla a otra durante la explicación de cómo ha sido realizado el proyecto, a continuación volvemos a repasar el funcionamiento de forma que sirva como manual de usuario de cómo funciona la aplicación que hemos diseñado.

Una vez que ejecutamos la aplicación en nuestro terminal, lo primero que nos aparecerá será una pantalla que nos permitirá introducir el nombre del artista sobre el que deseamos obtener información y dos botones que nos llevaran a la discografía o a los conciertos según corresponda.

Si pulsamos sobre el botón Discografía, nos llevará a una pantalla en la que se muestra el nombre del artista que estamos consultando y la lista de los discos editados por este artista. Para cada uno de los discos de la lista nos aparece en la parte izquierda una pequeña imagen de la portada del mismo y a la derecha el nombre del disco con su año de publicación y bajo qué sello discográfico fue publicado.

Desde esta pantalla, podremos hacer dos acciones. La primera sería pulsar sobre la imagen de la portada lo que hará que se abra una pantalla en la que se muestra la imagen de la portada del disco con mayor calidad. La otra acción que podemos

realizar en la pantalla de la discografía es pulsar sobre el disco que deseemos lo que hará que se abra una nueva pantalla en la que aparecerán las canciones que componen el disco.

En esta pantalla se muestran las canciones que componen el disco, en la parte superior se aparece la portada del disco y su información, y en la parte inferior la lista con el nombre de las canciones que componen el disco y dos botones. Si pulsamos el primero de ellos, nos llevara a una pantalla en la que podremos leer la letra de la canción que hayamos seleccionado. Y si pulsamos el botón Play además de poder leer la letra nos permite ver el video de Youtube más relevante que corresponde a la canción que hemos seleccionado.

Volviendo a la pantalla inicial, la otra opción que tenemos es pulsar el botón Conciertos. Al pulsarlo, nos aparecerá una pantalla que contiene la lista de los conciertos ordenados del más reciente al más antiguo. Para cada uno de los conciertos que aparece se mostrara el nombre descriptivo que identifica al concierto y su fecha y lugar de celebración.

Al pulsar sobre el concierto que queramos consultar, pasaremos a una nueva pantalla que es similar a la de las canciones que formaban parte de un disco con la única diferencia que en la parte superior en lugar de aparecer la información del disco aparecerá la del concierto elegido. Para cada una de las canciones de la lista también tendremos la posibilidad de visualizar su letra o ver el video.

6 Conclusiones

La conclusión en cuanto al desarrollo de aplicaciones web que extraen información de Internet mediante *web scraping*, considero que son de gran utilidad, puesto que internet existe mucha información que puede estar repartida por distintas fuentes y la posibilidad de tener en una misma aplicación información de distintos sitios web sin tener que visitar varias páginas hace que sea más sencillo de acceder a ella.

Además, ahora que la mayoría de las veces consultamos internet desde nuestros terminales móviles es más cómodo visualizar esos datos en una aplicación diseñada para ello que en la propia web porque muchas aún no están pensadas para ser leídas desde un dispositivo móvil. Por ejemplo, es más cómodo leer las noticias del periódico en la aplicación que este nos proporciona que en la propia página del periódico puesto que la aplicación está pensada para mostrar la información en los dispositivos móviles. Además, esta aplicación nos puede ofrecer más funcionalidades que la página visualizada en el navegador.

En cuanto a la aplicación que se ha desarrollado, creo que el resultado final cumple con lo que me planteé que debería hacer la aplicación cuando comencé a desarrollarla. Pero como toda aplicación es mejorable y tal y como expongo en el próximo capítulo ya tengo pensado una serie de mejoras que harán que ofrezca más funcionalidades que hasta ahora.

Por lo tanto, estoy bastante satisfecho con el resultado puesto que me sirve para los propósitos que quería conseguir cuando pensé en su diseño. Yo la uso bastante sobre todo para ver videos a la vez que leo la letra y de paso practico inglés. Además a las personas que se la he dejado para que la probaran les ha gustado y me han animado a seguir trabajando en ella.

Por último, en cuanto lo que me ha aportado personalmente el proyecto, ha sido el conocimiento adquirido en el análisis de páginas web y el desarrollo de aplicaciones móviles para Android, además que me ha motivado en seguir profundizando en el tema y ya no solo me planteo añadirle nuevas funcionalidades a esta aplicación, sino desarrollar alguna otra siempre que el tiempo libre me lo permita.

7 Futuras líneas de trabajo.

La aplicación desarrollada cumple los objetivos que deseaba conseguir cuando me planteé crearla, pero como toda aplicación, una vez que se comienza a usar siempre surgen nuevas funcionalidades que se podrían añadir. Algunas de las nuevas funcionalidades han sido sugeridas por personas que han probado la aplicación y que han aportado una visión más global de lo que podría ser útil para el usuario final.

A continuación, se detallan las nuevas que funcionalidades que han surgido durante las pruebas, algunas de las cuales se irán desarrollando para añadirlas en un futuro con el fin de que sea más completa:

- Visualizar además de la portada del disco seleccionado el resto de las imágenes del disco que nos proporciona la web de coveralia. Por ejemplo, trasera del disco, algunas páginas del libreto o la imagen xerografiada en el CD.

- Posibilidad de hacer zoom sobre las imágenes para poder ver aquellos detallas de la misma que se nos escapan a primera vista.

- Añadir el botón de compartir al igual que tienen otras aplicaciones para poder enviar la portada o la letra de la canción a otra persona por Whatsapp o publicarla en alguna de las redes sociales de las que usamos.

- Posibilidad de incluir un pequeño banner publicitario con la finalidad de sacarle alguna rentabilidad a la aplicación. Esta funcionalidad, que me fue sugerida por una de las personas que probó la aplicación, de momento está descartada puesto porque no es mi objetivo obtener ningún lucro con esta aplicación, sino simplemente ofrecer al usuario una aplicación que le pueda ser útil y agradable de manejar sin que tenga que aguantar que le vaya apareciendo publicidad en la plantilla.

- Incluir la búsqueda inteligente de Google puesto que en muchas ocasiones no conocemos de forma exacta el nombre del artista que deseamos buscar y esto sería de gran utilizar sobre todo para artistas extranjeros.

- Posibilidad de listar todas las canciones de un artista dado que muchas veces sabemos que tiene una canción pero no sabemos en qué disco aparece.

La aplicación tengo pensado subirla al Market de Android para que pueda ser descargada por aquellas persona que consideren que les pueda ser interesante tenerla instalada en su dispositivo móvil para consultar información musical.

Ya fuera del desarrollo de la aplicación diseñada para este proyecto, mi intención es seguir profundizando en el conocimiento del desarrollo de aplicaciones Android para dispositivos móviles con el fin de poder crear alguna otra aplicación que me apetezca desarrollar siempre a modo de hobbie.

Para ello, además de más de seguir leyendo manuales y ver videos de cursos en YouTube, me planteo apuntarme a algún curso online y de esta forma profundizar aún más en el conocimiento del desarrollo de aplicaciones Android.

Bibliografía

[sgoliver] http://www.sgoliver.net/blog/curso-de-programacion-android/. Curso de programación Android

[Naughton] "Manual de Java" Patrick Naughton

[wiki01] https://es.wikipedia.org/wiki/Web_scraping Articulo de Web scraping en Wikipedia.

[youtube01] https://www.youtube.com/watch?v=jSSMfRhi7SI Video curso programación Android

[school] http://es.schoolofdata.org/introduccion-a-la-extraccion-de-datos-de-sitios-web-scraping/ Introducción a la estraccion de datos mediante web scraping

[youtube02] https://www.youtube.com/watch?v=iokpfscL47o Video explicación de cómo hacer un API a partir de web scraping

[micayael] http://blog.micayael.com/2011/02/09/metodos-get-vs-post-del-http/ Utilización de los métodos GET y POST para peticiones HTTP.

[genbeta] http://www.genbeta.com/web/un-captcha-sin-captcha-google-ya-sabe-cuando-no-eres-un-robot Información del uso de reCAPTCHA

[developers] https://developers.google.com/youtube/v3/?hl=es Documentación en Google Developers del funcionamiento del YouTube Data API v3

[android01] http://developer.android.com/tools/publishing/app-signing.html Android: Signing Your Applications